2

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

EFFECT OF FIBER DIAMETER ON THE RELIABILITY OF COMPOSITES — AUTOMATED LASER DIFFRACTION IMPLEMENTATION

by

Jeffrey Scott Kunkel

December 1987

Thesis Advisor: Edward M. Wu

Approved for public release; distribution is unlimited.

A193 524

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)

6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School	6b OFFICE SYMBOL (If applicable) Code 67	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000		7b ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION	8b OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)		10 SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO	TASK NO	WORK UNIT ACCESSION NO.

11 TITLE (Include Security Classification)
EFFECT OF FIBER DIAMETER ON THE RELIABILITY OF COMPOSITES -- AUTOMATED LASER DIFFRACTION IMPLEMENTATION

12. PERSONAL AUTHOR(S)
KUNKEL, Jeffrey Scott

13a. TYPE OF REPORT Engineer's Thesis	13b TIME COVERED FROM ______ TO ______	14. DATE OF REPORT (Year, Month, Day) 1987, December	15 PAGE COUNT 118

16 SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB-GROUP

18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Fiber Diameter Measurement by Laser Diffraction
Effect of Stochastic Diameter on Composite Reliability

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

Composite failures are microscopically sequential and locally redundant. As a result, a composite structure reliability and its strength dependency on geometric size is intimately dependent on the statistics of fiber filament strength. A composite reliability model is needed to utilize such inherent materials redundancy in structural design. This investigation first establishes the important role of fiber diameter measurement in the characterization of fiber filament strength statistics and the composite reliability function, and second, implements the diameter measurement by laser diffraction. This method is automated and lends itself to industrial adoption for materials development, acceptance and quality control.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED □ SAME AS RPT □ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
22a NAME OF RESPONSIBLE INDIVIDUAL Professor Edward M. Wu	22b TELEPHONE (Include Area Code) (408) 646-3459	22c OFFICE SYMBOL 67Wt

83 APR edition may be used until exhausted
All other editions are obsolete

☆ U.S. Government Printing Office: 1986—606-24

Approved for public release; distribution is unlimited.

Effect of Fiber Diameter on the Reliability of Composites — Automated Laser Diffraction Implementation

by

Jeffrey Scott Kunkel
Lieutenant, United States Navy
B.S.E.E., United States Naval Academy, 1980
M.S.A.E., Naval Postgraduate School, 1987

Submitted in partial fulfillment of the
requirements for the degree of

AERONAUTICAL ENGINEER

from the

NAVAL POSTGRADUATE SCHOOL
December 1987

Author: ______________________
Jeffrey S. Kunkel

Approved by: ______________________
Edward M. Wu, Thesis Advisor

____
M. F. Platzer, Chairman, Department of Aeronautics

____
G. E. Schacher, Dean of Science and Engineering

ABSTRACT

Composite failures are microscopically sequential and locally redundant. As a result, a composite structure reliability and its strength dependency on geometric size is intimately dependent on the statistics of fiber filament strength. A composite reliability model is needed to utilize such inherent materials redundancy in structural design. This investigation first establishes the important role of fiber diameter measurement in the characterization of fiber filament strength statistics and the composite reliability function, and second, implements the diameter measurement by laser diffraction. This method is automated and lends itself to industrial adoption for materials development, acceptance and quality control.

TABLE OF CONTENTS

LIST OF FIGURES

TABLE OF SYMBOLS

A	amplitude of a diffraction pattern at a point P
A_i	amplitude of a single point light source
A_o	amplitude at the center of a diffraction pattern
b_n	ratio of $J_n(\alpha) / H_n^{(2)}(\alpha)$
d	distance between two point sources of light
D_f	diameter of a fiber
D_s	diameter of a slit
I / I_o	relative intensity of a diffraction pattern at a point P
$H_n^{(2)}(\alpha)$	Hankel functions of the second kind of order n
$J_n(\alpha)$	Bessel functions of the first kind of order n
K_o	constant for a specified λ
K_1, K_2	constants for a specified n
m	index of refraction for a fiber
n	number of a diffraction pattern minimum as indexed from the center of the pattern
P	a point far from a slit or a fiber
P_f	probability of failure
R_n, S_n	simplified coefficients for computing I / Io
s	distance from fiber or slit to the plane of the diffraction pattern
W	statistical distribution
x	distance in the plane of a diffraction pattern from the center of the pattern to a specified point
X	distance between the two MicronEyes™
$Y_n(\alpha)$	Bessel functions of the second kind of order n
α	Weibull shape parameter
β	Weibull scale parameter

δ	path difference between two point sources of light
γ	phase difference between two point sources of light
Γ	phase difference between two point sources of light that are D_s / 2 apart
λ	wavelength of light
μ	Gaussian mean
Φ	phase difference between the edges of a slit
σ	Gaussian standard deviation
θ	scattering angle

ACKNOWLEDGMENT

I would like to thank my wife Barbara for the support and assistance that she provided to me during the latter stages of my thesis. Her patience during the times of frustration and her words of encouragement when things looked bleak gave me the confidence to hurdle each obstacle as it arose. Barbara provided me with an ideal environment to study in, and I am grateful for that.

I would also like to express my gratitude to my thesis advisor, Professor Edward M. Wu. His enthusiasm and personal motivation stimulated my interest in the field of composite materials and challenged me to achieve more than I might otherwise have attempted. I was very appreciative of his open door policy and of his willingness to assist, however he could. I consider myself fortunate to have had Professor Wu as an instructor.

I. INTRODUCTION

Over the past decade, the advent of composite materials has radically altered the field of structural design. Where the designer previously had a finite number of materials with which to build, an infinite number of choices are now available. Where design deficiencies previously could only be corrected by modifying the geometry of a structure, adjusting the material of the structure may now correct the problem. Where peculiar loading or stability problems previously resulted in massive, overdesigned structures, the technology now exists to design light and efficient ones. In short, the ability to design a material to meet precise specifications has unshackled the designer from previous conventional limitations and provided much greater flexibility.

The capability to design the material for an application does not come without some costs. The ability to exploit the directional characteristics of composite materials also requires that many more variables be optimized during the design phase of a project. The manufacturing process is much more complex, and structural repair requires different approaches. The failure mechanisms of composite materials are microscopically sequential, and reliability, life, and strength estimates for complicated structures are difficult to predict. Finally, structures can be very expensive to construct, frequently precluding extensive destructive testing.

Composite failures are microscopically sequential and locally redundant. In order to utilize microscopic redundancy of composites in structural design, a realistic reliability model is needed. The chain-of-bundles model [Refs. 1 and 2] relates fiber filament failure (which is serial) to composite failure (which is locally parallel). With this model, if statistical parameters for the fiber filaments are accurately known, the statistical strength parameters for the composite can be determined. With the composite statistical strength model, the composite reliability dependency on the structural dimensional size and service stress level can be quantified.

Traditionally, the statistical strength parameters for fibers are measured in terms of failure loads. This investigation first focused on the importance of accounting for the diameter variations in the statistical characterization of fibers. An analysis of the stochastic interaction of the randomness in failure load and in fiber diameter demonstrated the importance of fiber diameter measurement in the resulting composite reliability characterization. Second, the implementation of such diameter measurements was accomplished by an automated, highly accurate process involving the computer digitization and processing of a laser diffraction pattern from a single fiber, which is presented and discussed.

II. BACKGROUND

Structural design of composite material structures uses the concept of stress, or force per area, as a method of parameters reduction. Loads can be measured very precisely, but accurately measuring the diameters of the fibers that typically make up a composite ply is difficult. Fibers frequently have diameters on the order of 10 μm, or approximately ten times finer than a human hair.

To date, it has been common to use an average diameter for calculations. However, extreme value parameters such as strength, life, and ultimate stress are more correctly described by a statistical distribution of values. As a result, paradoxically, structural design and analysis are based on stress; whereas the materials parameters (strength) input into the analysis are based on force. This inconsistency may lead to erroneous and perhaps nonconservative designs. For example. the statistical parameters for fiber strength (mean, variability) in terms of stress may change when the statistical scatter in fiber diameter is accounted for. If the statistical distribution of the diameter of the fibers could be used in calculations instead of the averaged values, important parameters such as life, strength, and stress could be modeled much more accurately than is currently possible.

A. LIFE

Service lives of composite material structures are frequently measured in large time units of mean life, such as thousands or millions of years. Large mean lives is a consequence of large life variability, which is typically three decades or more. A large mean life in millions of years is needed to assure high reliable life in tens of years. Since the fibers for many high performance composites have been synthesized only recently, no service experiences are available. Obviously, it is impractical to establish a statistical distribution of life expectancy in real time, so a suitable method of time acceleration needs to be established. Typically, the stress or strength of a specimen is related to the life of the specimen, where high strength is indicative of long life and low strength is indicative of short life. Phoenix and Wu [Ref. 3] showed that the level of stress used for life measurement is critical and that a very small change in stress may change the life estimate by an order of magnitude or more. Knowing the diameter of the fibers in a composite materials is thus crucial if accurate life estimates are desired.

B. RELIABILITY

Reliability is the probability that a structure will not fail. Usually, reliability is expressed in terms of failure stress or life expectancy. For small structures, a statistical distribution of failures can be experimentally obtained, an appropriate curve fit to the data, and reliability can be quantified. However, with large structures, statistics through destructive testing is cost prohibitive.

One method of quantifying the reliability of large structures involves a sequence of probabilistic mathematical models. First, a relationship between a single fiber and a single composite ply (many fibers) is postulated. Similarly, a relationship between a single ply and a large structure can be established. In this way, the distribution of characteristic parameters for single fibers can be extrapolated to enormous structures and probabilistic reliability estimates can be made to useful precision. Again, because stress is dependent on the cross sectional area, or diameter, of the fiber, accurate diameter estimates are necessary. [Refs. 4-6]

III. DIAMETER AS A STOCHASTIC VARIABLE

The structural design process typically uses stress, strain, and their combination, strength, as primary variables. Strictly speaking, strength parameters in terms of loads should not be used in stress analysis without accounting for the probabilistic interaction of the random variables. Strain is directly observable when tensile failure load is measured. However, because stress cannot be measured directly, laboratory analyses must measure force and area and calculate the stress using an equation (stress equals force divided by area). If the cross sectional area is a deterministic (rather than stochastic, or random) variable, the statistical strength based on stress and that based on load will differ only by a constant. If the cross sectional area is not deterministic, there will be a nonlinear relationship between stress and load distributions.

A. OBSERVATION BY COMPUTER SIMULATION

Appendix A is a discussion of the analytical interaction of the random variables failure load, cross sectional area, and failure stress. Here, computer simulation is used to illustrate the characteristic relationship between these variables. For a given set of fiber samples, both the failure loads and the cross sectional areas will have a statistically distributed range of values. On the physical grounds thar a filament fails by the weakest link process, a two parameter Weibull distribution [Ref. 7] can be used to model the failure loads. The appropriate distribution function to best model the diameters of the fibers will vary with the manufacturing process. It may be argued that drawn fibers will have an upper bound diameter limited by the oriface. Other factors that may affect the diameter of the fibers include the rate of extrusion, the ambient temperature, the viscosity of the fiber material, and the handling and loading of the fibers after fabrication. Rational modeling of the distribution for fiber diameters to account for these effects are outside the realm of this investigation. For illustration herein, a truncated normal distribution (diameter always positive) was used to represent the range of diameter values.

A series of computer simulation experiments was conducted to simulate independent distributions of fiber diameter and fiber failure load using truncated normal and Weibull distributions respectively. Standard deviations for the fiber diameter distribution of 5%, 10%, and 20% were used. The stress was calculated for each sample and Weibull parameters were determined for the stress distributions [Ref. 8: pp. 426-7]. Table 1 summarizes the results of the simulations. The shape parameter α for stress was always less than that for load, and the magnitude of the difference between the two values was a function of the standard deviation of the fiber diameter distribution. Figure 1 illustrates the apparent relationship observed via numerical simulation between fiber diameter standard deviation and the change in the shape parameter α.

TABLE 1. RESULTS OF COMPUTER SIMULATION

Std Dev. of Diameter, σ	α_{load}	α_{stress}	$(\alpha_{load} - \alpha_{stress})/(\alpha_{load})$
0.05	4.979	4.946	0.0066
0.10	4.947	4.855	0.0185
0.20	5.033	4.496	0.0670

The shape parameter of the Weibull distribution is approximately inversely proportional to the variability. The physical consequences of characterizing failure strength by load (i.e., not accounting for diameter variation) is an underestimate of the variability of the intrinsic strength. That is, the strength variability measured by load is lower than the actual strength variability measured by stress (accounting for diameter variation). The effect of such error in the shape parameter estimate on the reliability of a large structure is discussed in the following section.

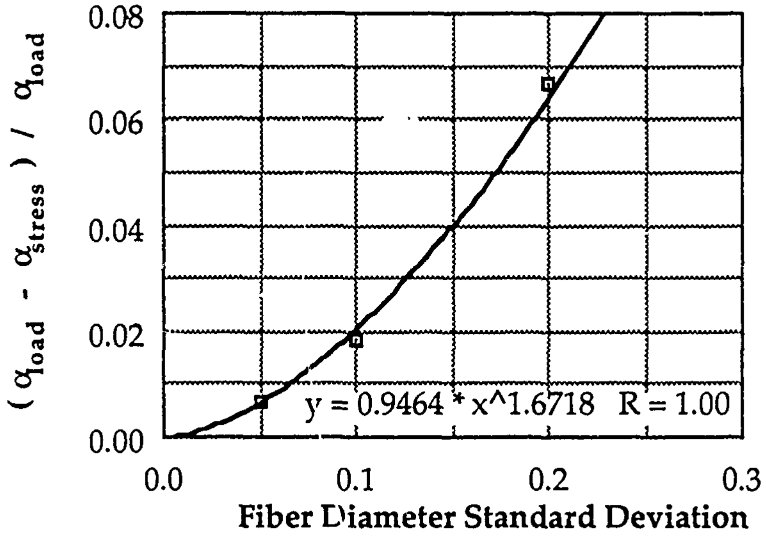


Figure 1. Effect of Fiber Diameter Standard Deviation on Shape Parameter α

B. ESTIMATING BOUNDS FOR RELIABILITY

Substantial amounts of research have been published on the subject of estimation of composite material reliability from the statistical properties of fiber strength. Phoenix and Smith [Ref. 5] summarized the work of Harlow and Phoenix [Refs. 1, 2, 9-11] and others and provided three methods of estimating the strength of fibrous composite materials using a chain-of-bundles model with local load-sharing between adjacent fibers. One of these results is used in this investigation; it estimated the composite statistical strength distribution W(x) with

$$W(x) = \min F^{[k]}(x), \qquad x \geq 0; \; k = 1, 2, 3, \ldots \tag{1}$$

where

$$F^{[k]}(x) = 1 - \exp\{-d_k (x / \beta)^{k\alpha}\}, \qquad x \geq 0 \tag{2}$$

$$d_k = d_k(\alpha) = 2^{(k-1)} (K_1 K_2 \ldots K_{k-1})^{\alpha} \tag{3}$$

$$K_r = 1 + r / 2, \qquad r = 1, 2, 3, \ldots \tag{4}$$

[Ref. 5]

k = 1 is the condition that when one filament fails, the entire structure fails; the representation of a single filament. Therefore, given the statistical representation of a single filament, i.e., $F^{[1]}(x)$, the probability of composite failure can be estimated from W(x) by using Equation 1.

The relationship between the single fiber strength distribution $F^{[1]}(x)$ and the composite structure distribution W(x) is illustrated in Figure 2 (in Weibull probability coordinates). Three important relations are observed.

(1) The composite distribution is no longer linear in the Weibull probability space (i.e., the composite failure is no longer serial).

(2) The slope (shape parameter) of the fiber strength distribution controls the slope of the composite distribution at the upper tail.

(3) Small changes in the upper tail slope leads to amplified "rotation" of the lower tail slope.

The third observation is of greatest structural importance. The reliability of Figure 2 is normalized for the physical dimension of the interface ineffective length (approximately 10 D_f or 50 μm). The equivalent size for a rocket motor case (measured in this scale) is on the order of 10^{25}. That is, the portion of the function W(x) which controls the reliability of large structures is around $P_f = 10^{-25}$. There, operating at small values of (x / β) or, equivalently, with large structures, will necessitate accurate estimates of α if accurate distributions W(x) are required.

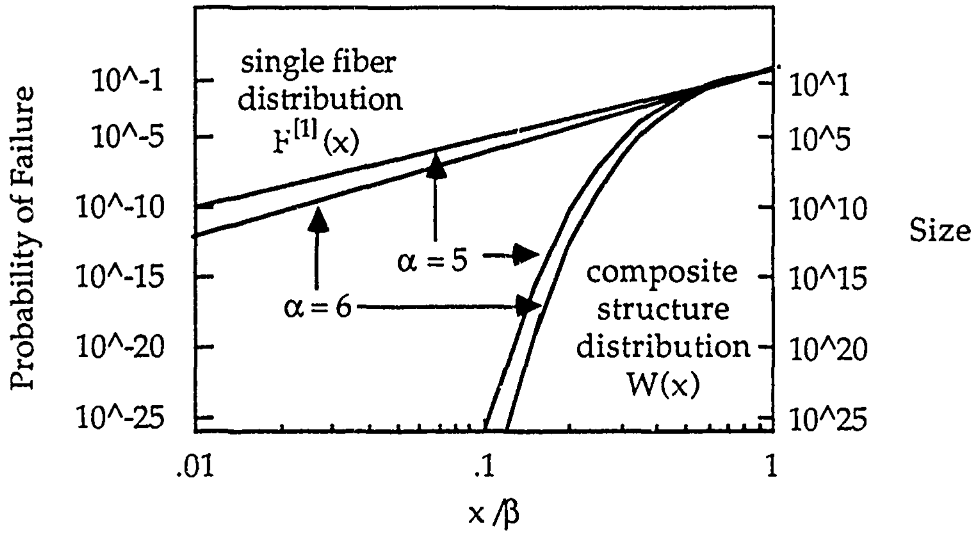


Figure 2. Sensitivity of Composite Material Reliability to Shape Parameter α

C. EFFECT OF THE DIAMETER DISTRIBUTION ON RELIABILITY

The stress distribution based on the results of the computer simulation for a 20% standard deviation in fiber diameter is plotted in Figure 3. Clearly, the use of a stochastic diameter variable rather than a deterministic one changes the value of (x / β) for a given structure size and desired reliability. If the standard deviation of the diameter of the fibers used to build a structure with fibrous composite materials was 20%, the structure would be potentially unsafe, because at a given structural load (x / β) the probability of failure predicted using the inappropriate parameter, load, is in fact lower.

Hence, it is numerically demonstrated that ignoring the statistical variation in fiber diameter can lead to an erroneous shape parameter slope based on load, which in turn "rotates" the upper tail slope of the composite distribution, causing a nonconservative error in the lower tail. Because the lower tail is divergent, as the structure becomes larger, the error introduced by the ignoring the fiber diameter becomes more severe.

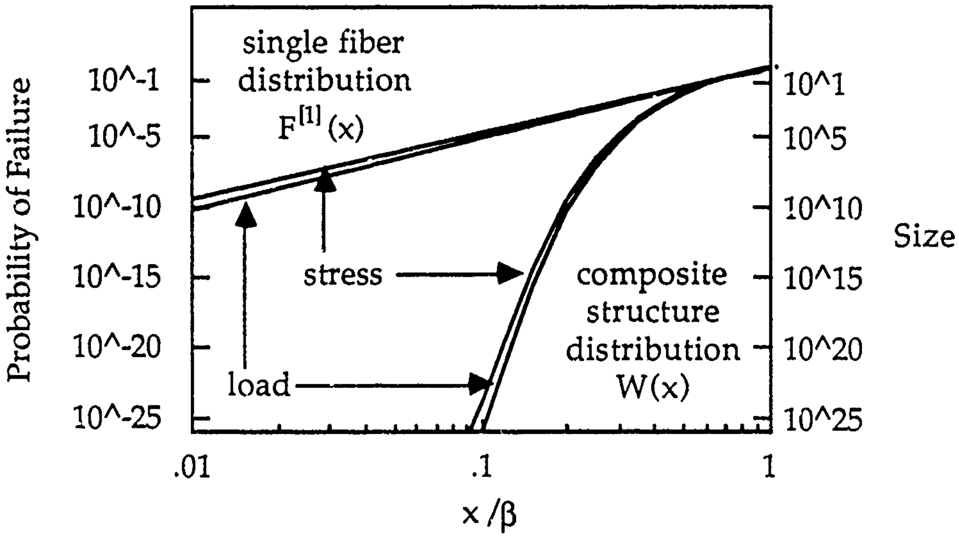


Figure 3. Reliability Estimate for a Fiber Diameter Standard Deviation of 20%

IV. DIAMETER MEASUREMENT

Several methods of measuring the diameters of very small fibers are available. Each method has advantages and disadvantages. With the goal of transferring fiber diameter measurement from the laboratory to quantify assurance in materials and manufacturing, this thesis work advocates the use of an automated approach using laser diffraction.

A. METHODS AVAILABLE

Measuring the diameter of a fiber that is on the order of 10 μm requires careful consideration. Some methods that provide precise results may damage the specimen, thus trivializing the measurement process.

1. Optical Microscope

The optical microscope is fast, simple, inexpensive, is commonly available, and can produce precise results. However, it requires direct human manipulation and interpretation, has limited resolution, and can be fatiguing for the operator if large numbers of measurements are to be made.

2. Electron Microscope

The electron microscope produces very precise results, has excellent resolution, is less dependent on operator skill, and is not fatiguing for the operator. However, the equipment is very expensive, is not commonly available, and the process requires pretreating the samples with a conductive coating that may alter the strength of the sample.

3. Photoconductive Cell

Previous thesis research by Bennett [Ref. 12] demonstrated that locating the minimums of a diffraction pattern using a photoconductive cell and applying these minimums to the classic Fraunhofer diffraction pattern theory (see Appendix B) yielded reasonable results. The method has the advantages of being nondestructive to the sample and of using inexpensive equipment. However, the precision of the measurement is dependent on the skill of the operator, the process is tedious, and some postprocessing calculation is required.

B. AUTOMATED APPROACH USING LASER DIFFRACTION

The diffraction theory for a slit and for a fiber are discussed in detail in Appendix B. These theories have been applied together with some inexpensive equipment to produce an automated system for measuring the diameter of a fiber to within 1% in less than 30 seconds.

1. Hardware

The following pieces of equipment were used in the automated diameter measuring system.

a. MicronEye™

Each MicronEye™ system consists of a IS32 OpticRAM chip mounted behind a camera lens and connected to a computer interface device. Briefly, the IS32 OpticRAM chip is a conventional computer memory chip with its protective cover removed, exposing its rectangular array of light sensitive pixels. These pixels can be interfaced with a computer. The specific technical data for the IS32 OpticRAM, the heart of the MicronEye™ system, is contained in Reference 13. The horizontal resolution is better than 9 μm per pixel, and the chip is sensitive to light with wavelengths up to near the ultraviolet range. Two MicronEye™ systems were used.

b. Macintosh™ Computer

The Macintosh™ computer was selected for its high resolution graphics capability, its ease of interface with the MicronEye™, its ease of use by the operator, its speed of calculation, and its ability to handle a large number of significant digits during calculations.

c. Helium-Neon Laser

A low power 1 mW Helium-Neon laser was used as a collimated light source. The wavelength of a Helium-Neon laser is 0.6328 μm.

d. Digital Caliper

A digital caliper was used to measure the distance between the two MicronEyes™. The caliper was able to measure distances within 0.01 mm.

2. Software

The Macintosh™ application CALIPER was written as part of this thesis research. The source code is contained in Appendix C. The programming language C was chosen because of its powerful graphics capabilities and because of its speed of execution on the Macintosh™.

3. Equipment Setup

Figure 4 is a schematic diagram of the system. The laser, fiber support stand, and MicronEyes™ were all mounted on a stiff table using rigid rails. The MicronEyes™ were mounted to tables on the rails that had integral micrometers for precise adjustment. The MicronEyes™ were connected to the Macintosh™ computer via interface devices. The digital caliper was mounted to the MicronEye™ supports. The fibers were mounted to cardboard holders that were in turn clamped in a support stand. The support stand was mounted to a table on the rails that had micrometers to allow precise adjustment in the directions parallel and perpendicular to the laser beam. Figure 5 shows how a fiber was mounted for measurement.

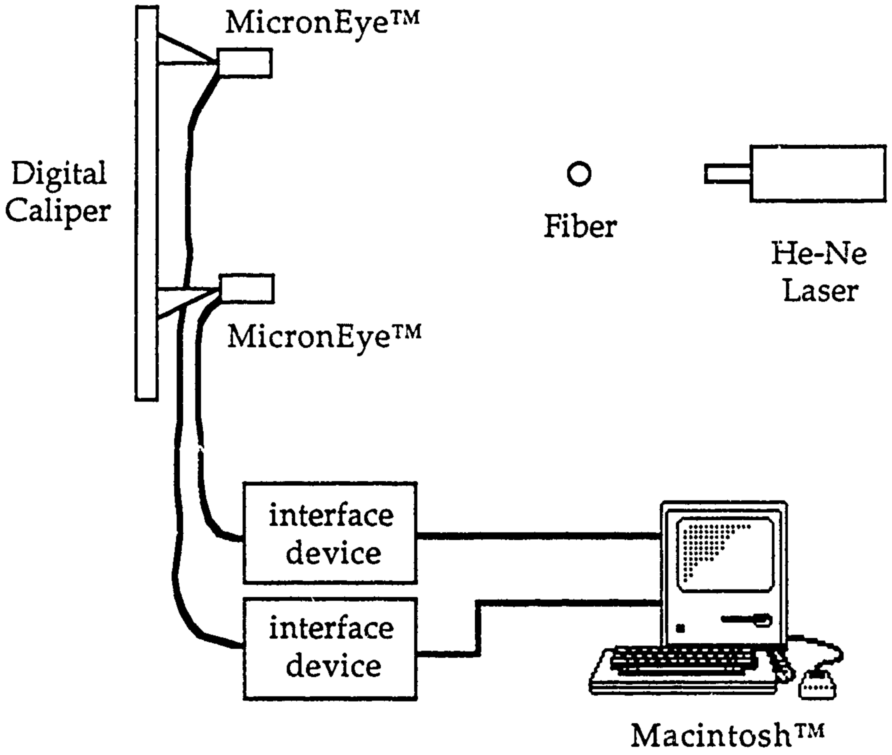


Figure 4. Schematic of an Automated Diameter Measuring System

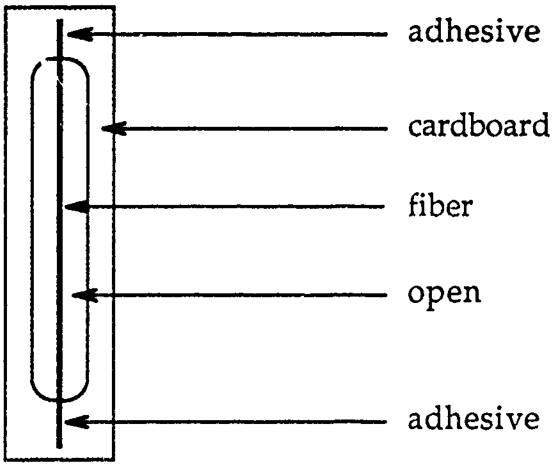


Figure 5. Mounting a Fiber Sample for Measurement

4. Procedure

A cardboard mounted fiber was placed in the support stand and the stand was adjusted to place the fiber in the center of the laser beam. The MicronEyes™ were then centered on diffraction pattern minimums symmetrically about the centerline of the laser beam. The system geometry was input to the CALIPER program by the operator at the beginning of program execution and after any movement of the components. Typically, once the equipment was positioned, many fibers with similar diameters could be measured without requiring readjustment of the equipment.

Exposure times were adjusted until distinct minimums were located on the Macintosh™ screen for both MicronEyes™ (one diffraction pattern at a time). Figure 6 shows the desired display for one image. Again, once the exposure settings were determined for one fiber, they were applicable for all fibers with similar diameters.

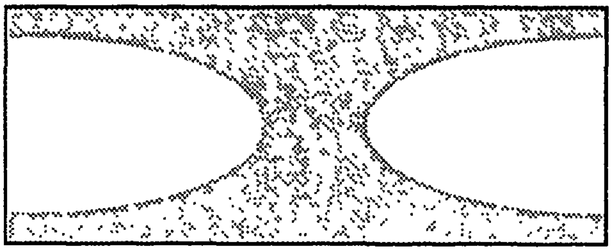


Figure 6. Centering Diffraction Minimums

After all data had been entered and all equipment adjusted, the program would digitize the diffraction patterns, process them, and report the diameter of the fiber by three methods: using just the left-hand MicronEye™, using just the right-hand MicronEye™, and using an average of the two results.

5. Computer Algorithm

If the minimums of the diffraction pattern of a fiber could be accurately located and measured, then the methods outlined in Appendix B could be applied and the diameter recovered with very high precision and accuracy. However, to locate these minimums exactly would require many exposures, and each exposure would require interaction with the operator. A faster and only slightly less accurate method uses only one exposure per MicronEye™.

It is known that the minimum lies between points A and B on the diffraction pattern shown in Figure 7. The minimum is very close to half way between points A and B. If the diameter is calculated under the assumption that the minimum is located at point A, the result will be too small. If the diameter is calculated using point B as the minimum, the result will be too large. If the results are averaged, the result is very close to the actual diameter. Computer simulation using this method resulted in errors of less than 0.1%

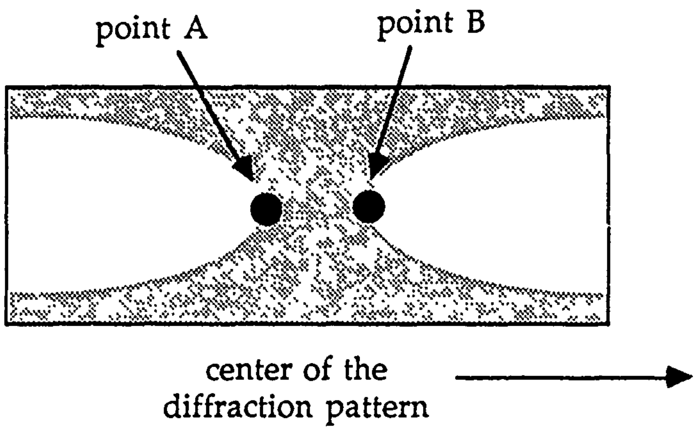


Figure 7. Approximating the Diffraction Pattern Minimum

6. Finding the Center of the Diffraction Pattern

The methods of Appendix B require the scattering angle θ to be known. The only practical method of measuring this angle is by measuring the distance s from the fiber to the plane of the MicronEyes™ and the distance x from the centerline of the pattern to the point where the angle is desired to be known. Figure 8 illustrates this geometry.

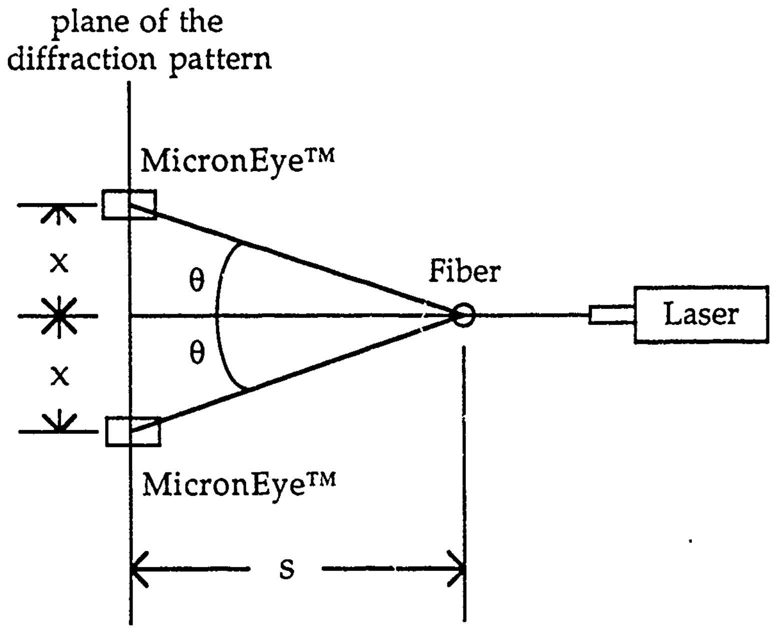
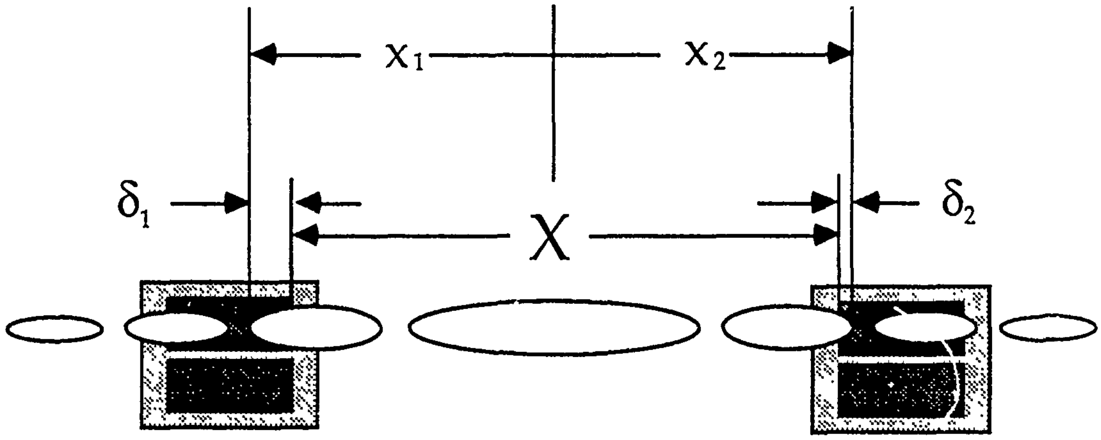


Figure 8. System Geometry

Small variations in the distance s will be negligible if s is chosen large with respect to x. However, small variations in x will be significant. Two MicronEyes™ were used to improve the resolution of the measurement of the distance x. Figure 9 shows how using two MicronEyes™ can locate the diffraction pattern center automatically, thereby reducıng the error in measuring the distance x.

Storch [Ref. 14] advocated optimizing the geometry of Figure 8 to yield two diffraction pattern minimums per MicronEye™. This approach was not used for several reasons. First, an automatic diffraction pattern centering scheme was deemed preferable to a manual one. Second, the resolution of the measurement of the angle θ was improved by using a larger percentage of the MicronEye™ sensor area. Third, by allowing a larger distance s between the fiber and the MicronEye™, sensitivity to errors in measuring s could be reduced. A larger distance s also allowed observation of small order diffraction pattern minimums (n = 1, 2) further from the very intense center of the diffraction pattern.



$$x_1 + x_2 = X + \delta_1 + \delta_2$$

$$\text{note that} \quad x_1 \neq X/2 + \delta_1$$

$$x_2 \neq X/2 + \delta_2$$

Figure 9. Locating the Center of the Diffraction Pattern

7. Summary of the Automated Approach

The approach presented above is fast, inexpensive, automatic, simple to operate, and very accurate. By properly selecting a laser based on the range of diameters to be measured (as discussed in Appendix B) fibers can be measured to within 1% or better over a range of diameters from less than 1 µm to more than 50µm.

V. RESULTS

The results for the two sections of this thesis research are summarized below.

A. DIAMETER AS A STOCHASTIC VARIABLE

It was demonstrated that statistical parameters of composites are relatable to the statistical fiber strength parameter by the chain-of-bundles model [Refs. 1 and 2]. Treating fiber diameter as a stochastic variable in reliability computations for composite materials consistently predicted weaker structures than when diameter was treated as a deterministic variable. Representing fiber diameter with a truncated Gaussian distribution and failure load with a two parameter Weibull distribution yielded a failure stress distribution that was accurately modelled by a two parameter Weibull distribution. Small variance in the shape parameter α for the distribution of a single fiber was seen to have a large effect on reliability predictions both for lightly loaded structures and for very large structures.

B. DIAMETER MEASUREMENT

By properly selecting a laser based on the range of diameters to be measured, the automatic system presented can measure fibers to within 1% or better over a range of diameters from less than 1 μm to more than 50 μm. Using an error function to directly relate Fraunhofer slit diffraction results to Kerker's [Ref. 15: p. 255] method for calculating the diffraction pattern of a fiber produced accurate results and significantly reduced computation time and memory requirements.

VI. CONCLUSIONS

Composite structural reliability depends strongly on the strength variability, or shape parameter, for the consistent single fiber filament. Very large composite structures with high required reliability levels are especially sensitive to the shape parameter of the fiber and its accurate estimation. The shape parameter α for failure stress depends on the stochastic interaction of both the failure load and the fiber diameter. Obtaining a statistical distribution of values for fiber diameter will necessitate an automatic, highly accurate measuring system. The presented system consisting of a Macintosh™ computer, two MicronEye™ light sensitive computer chips, and a low power Helium-Neon laser is an economical method of providing very accurate fiber diameter measurements, and hence, very accurate estimates of α. Most importantly, this method is automated and lends itself to industrial adoption for materials acceptance and quality control.

APPENDIX A. PROBABILISTIC INTERACTION OF MULTIPLE RANDOM VARIABLES

Filament strengths are traditionally measured in terms of failure loads and the statistical parameters are defined in terms of load. Structural design and analysis are operated in terms of stress. Therefore strength parameters in terms of load should not be used in stress analysis without accounting for the probabilistic interaction of the random variables. This is acceptable only if the area is a deterministic variable.

The following is an outline of how the distribution function $F_S(s)$, a function of the stochastic functions $f_D(d)$ and $F_P(p)$, can be obtained.

1. VARIABLE DEFINITIONS

The following variables are hereby identified and defined:

P	- Random (stochastic) failure load
p	- realized failure load
A	- Random area defined in terms of diameter D
a	- realized area
D	- Random diameter
d	- realized diameter
S	- Random failure stress
s	- realized failure stress
F_P	- Two parameter Weibull distribution
F_D	- Truncated normal distribution
$F_X(x)$	- Cumulative Distribution Function (CDF) for random variable X
$f_X(x)$	- Probability Density Function (PDF) for random variable X
α, β	- Parameters used to describe a Weibull statistical distribution

μ, σ - Parameters used to describe a Gaussian statistical distribution

2. IMPORTANT RELATIONSHIPS

$$S \equiv P / A \tag{A.1}$$

$$A = \pi D^2 / 4 \tag{A.2}$$

$$F_P(p) \equiv \Pr\{P \leq p\}, \qquad 0 \leq p \leq \infty \tag{A.3}$$

$$F_D(d) \equiv \Pr\{D \leq d\}, \qquad 0 \leq d \leq \infty \tag{A.4}$$

in general

$$F_X(x) \equiv \Pr\{X \leq x\} \tag{A.5}$$

$$f_X(x) \equiv \Pr\{X = x\} \tag{A.6}$$

3. GOAL

Equation A.1 and Equation A.2 relate failure stress S to failure load P and cross sectional area A. If P and A are deterministic variables, then calculating S is straightforward. If A is deterministic and P is stochastic, then

$$F_S(s) = F_P(p) / A \tag{A.7}$$

which is again straight forward. However, if P and A are both stochastic variables, then the division operator has no meaning. In other words,

$$F_S(s) \neq F_P(p) / F_A(a) \tag{A.8}$$

because the term [$F_P(p) / F_A(a)$] is undefined. Thus, the goal is to obtain an expression for $F_S(s)$, an unknown, in terms of known quantities.

4. OUTLINE

If a series of many experiments were conducted that measured both the failure load and the cross sectional area of a fiber, the failure stress for each fiber could be determined If matching pairs of failure load P and cross sectional area A were plotted, the failure stress S would be represented by the slope of a line drawn from the origin to the point. See Figure A1. The cumulative distribution function (CDF) for failure stress, $F_S(s)$, may be thought of as the sum of all the slopes that are less than s; this defines the domain of integration

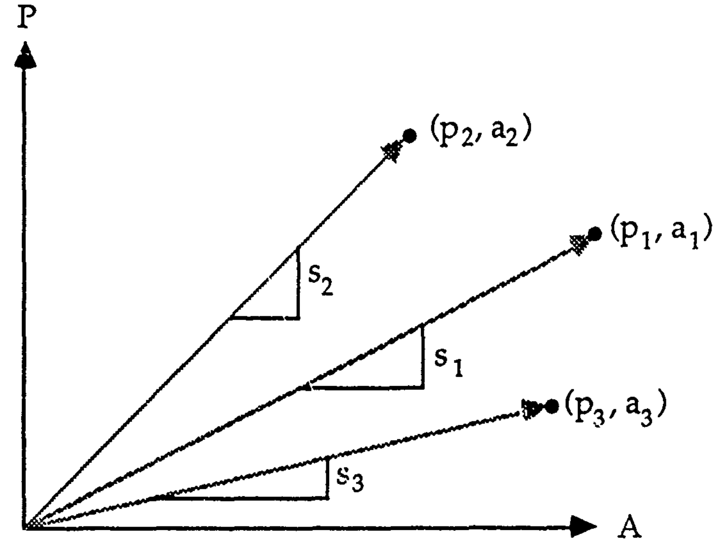


Figure A1. Stress is the Slope of a Line Drawn From the Origin

and is illustrated in Figure A2. Thus, $F_S(s)$ is the shaded area under the curve in Figure A2.

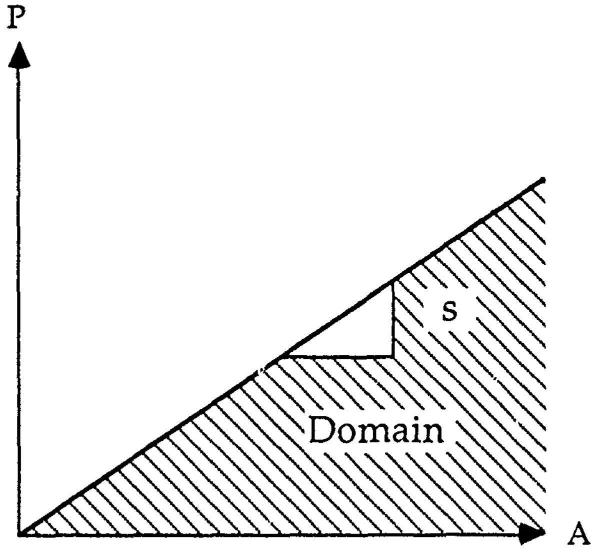


Figure A2. Illustrating the Domain of Integration

In equation form

$$F_S(s) = \int_{y=0}^{y=\infty} \int_{x=0}^{x=ys} f_{P,A}(x,y)\, dx\, dy \qquad \text{(A.9)}$$

where $f_{P,A}(p,a)$ is the joint probability distribution of P and A. If P and A are independent variables, then

$$f_{P,A}(p,a) = f_A(a)\, f_P(p) \qquad \text{(A.10)}$$

Combining Equation A.9 and Equation A.10

$$F_S(s) = \int_{y=0}^{y=\infty} \int_{x=0}^{x=ys} [\, f_A(y)\ f_P(x)\,]\ dx\ dy$$

$$= \int_{y=0}^{y=\infty} f_A(y)\ [\int_{x=0}^{x=ys} f_P(x)\ \ dx\,]\ dy \qquad \text{(A.11)}$$

However,

$$F_P(p) \equiv \int_{x=-\infty}^{x=p} f_P\,(x)\ dx = \int_{x=-\infty}^{x=0} f_P(x)\ dx + \int_{x=0}^{x=p} f_P(x)\ dx$$

$$= \int_{x=0}^{x=p} f_P(x)\ dx \quad \text{(all loads presumed positive)} \qquad \text{(A.12)}$$

So, Equation A.11 and Equation A.12 give

$$F_S(s) = \int_{y=0}^{y=\infty} f_A(y)\, F_P(ys)\, dy \qquad \text{(A.13)}$$

Also, note that

$$f_A(a) \equiv \Pr\{\, A = a \,\} = \Pr\{\, (\pi\, D^2 / 4) = a \,\} = \Pr\{\, D^2 = (4\, a / \pi) \,\}$$

$$= \Pr\{\, D = (4\, a / \pi\,)^{0.5} \,\} = f_D(\,(4\, a / \pi)^{0.5}\,) \qquad \text{(A.14)}$$

The fact that $D \geq 0$ eliminates negative roots.

Choose specific distribution functions to represent D and P.

$$F_P(p) \qquad = 1 - \exp\,[\, -\, (p / \beta)^{\alpha}\,], \qquad p \geq 0$$

(A.15)

where α and β are shape and scale variables, respectively (a two parameter Weibull distribution.)

$$f_D(d) \qquad = \exp\left[-(d-\mu)^2/(2\sigma^2)\right] / \left(\sigma(2\pi)^{0.5}\right), \quad d \geq 0 \qquad \text{(A.16)}$$

where μ is the mean and σ is the standard deviation (a truncated Gaussian distribution). Equation A.14 and Equation A.16 combine to give

$$f_A(a) = f_D((4a/\pi)^{0.5}) = \exp\left[-((4a/\pi)^{0.5}-\mu)^2/(2\sigma^2)\right] / \left(\sigma(2\pi)^{0.5}\right) \qquad \text{(A.17)}$$

Equation A.13, Equation A.15, and Equation A.17 can now be combined to give

$$F_S(s) = \int_{y=0}^{y=\infty} \frac{1}{\sigma(2\pi)^{0.5}} \exp\left[\frac{-((4y/\pi)^{0.5}-\mu)^2}{2\sigma^2}\right]$$

$$\left[1-\exp\{-(ys/\beta)^{\alpha}\}\right] dy \qquad \text{(A.18)}$$

Equation A.18 can be integrated numerically for any $s \geq 0$ if the four constants α, β, μ, and σ are specified. The constants α and β come from fitting a two parameter Weibull distribution to the failure load data. The constants μ and σ come from fitting a truncated normal distribution to the fiber diameter data.

5. SUMMARY

The CDF for failure stress S of a fiber can be obtained from experimental data using stochastic representations of failure load P and fiber diameter D by using Equation A.18 if P and D are assumed to be independent variables. A truncated Gaussian distribution was used for illustration purposes to represent the fiber diameter, but the method can be easily extended to other statistical representations.

APPENDIX B. LASER DIFFRACTION THEORY

This appendix discusses in detail the laser diffraction theories that were used during this investigation. The diffraction of a slit, the diffraction of a fiber, and a method of relating the two patterns are outlined and described in detail. Finally, the utility of these theories and their application are illustrated.

1. FRAUNHOFER DIFFRACTION

Characteristic diffraction patterns are produced by a sheet of laser light passing through a long slit when the wavelength of the laser light and the diameter of the slit are of the same order of magnitude. The Fraunhofer diffraction theory is well documented in most Physics textbooks. The following discussion was taken substantially from Tipler [Ref. 16: Chapter 25].

If the light that passes through the slit of diameter D_s is represented by N equally spaced point sources of equal amplitude light, then the diffraction pattern at any point far from the slit can be obtained by the vector summation of the contributions from each source. Consider Figure B1. If the distance to the point P where the pattern is being calculated is sufficiently far from the slit so that the rays from any two sources are essentially parallel, then the path difference between any two sources is

$$\delta = d \sin \theta \tag{B.1}$$

where d is the distance between the sources. The phase difference between the two sources is then

$$\gamma = 2 \pi \delta / \lambda = 2 \pi d \sin \theta / \lambda \tag{B.2}$$

The phase difference between any two sources that are $D_s / 2$ apart is

$$\Gamma = 2 \pi d \sin \theta / \lambda = 2 \pi [(D_s / 2) \sin \theta] / \lambda = \pi D_s \sin \theta / \lambda \tag{B.3}$$

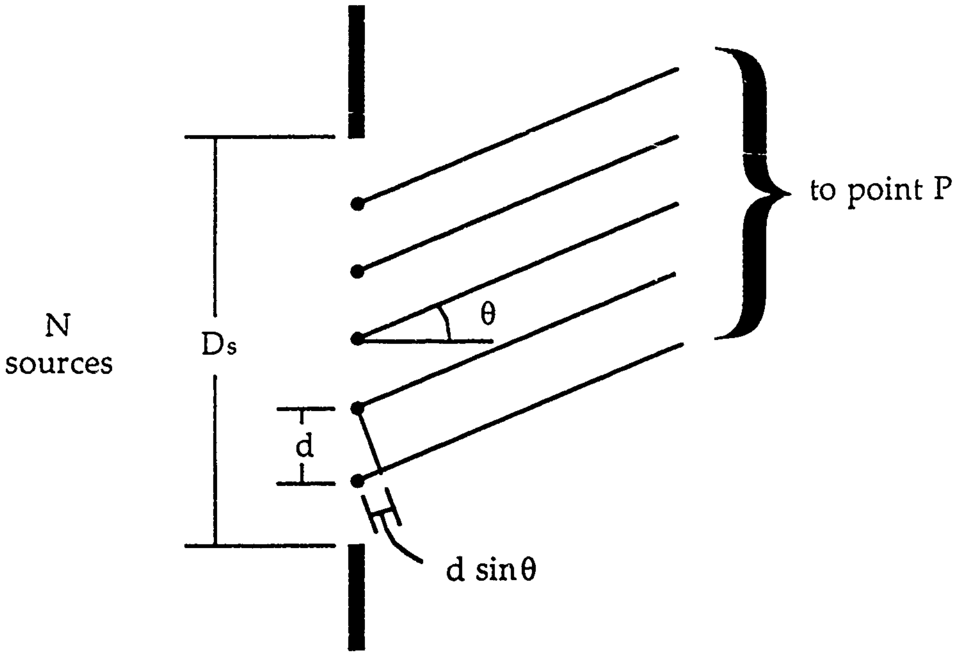


Figure B1. Modelling Fraunhofer Slit Diffraction

If $\Gamma = \pi$, then for every source contributing to the diffraction pattern there is another source that is exactly 180° out of phase with it and will exactly cancel the contribution. The result is a minimum of the diffraction pattern, as illustrated in Figure B2. Letting $\Gamma = \pi$ in Equation B.3 produces a more useful expression for a minimum of the diffraction pattern

$$\sin\theta = n\lambda / D_s \qquad n = 1, 2, 3, \ldots \qquad \text{(B.4)}$$

where n is the number of the minimum as indexed from the center of the diffraction pattern.

If the amplitude of each individual source is A_i, then the magnitude of the diffraction pattern at any point P far from the slit and offset from the centerline by an angle θ can be calculated. Let Φ be the phase difference between the slit edges. Equation B.2 then shows that

$$\Phi = 2\pi D_s \sin\theta / \lambda \qquad \text{(B.5)}$$

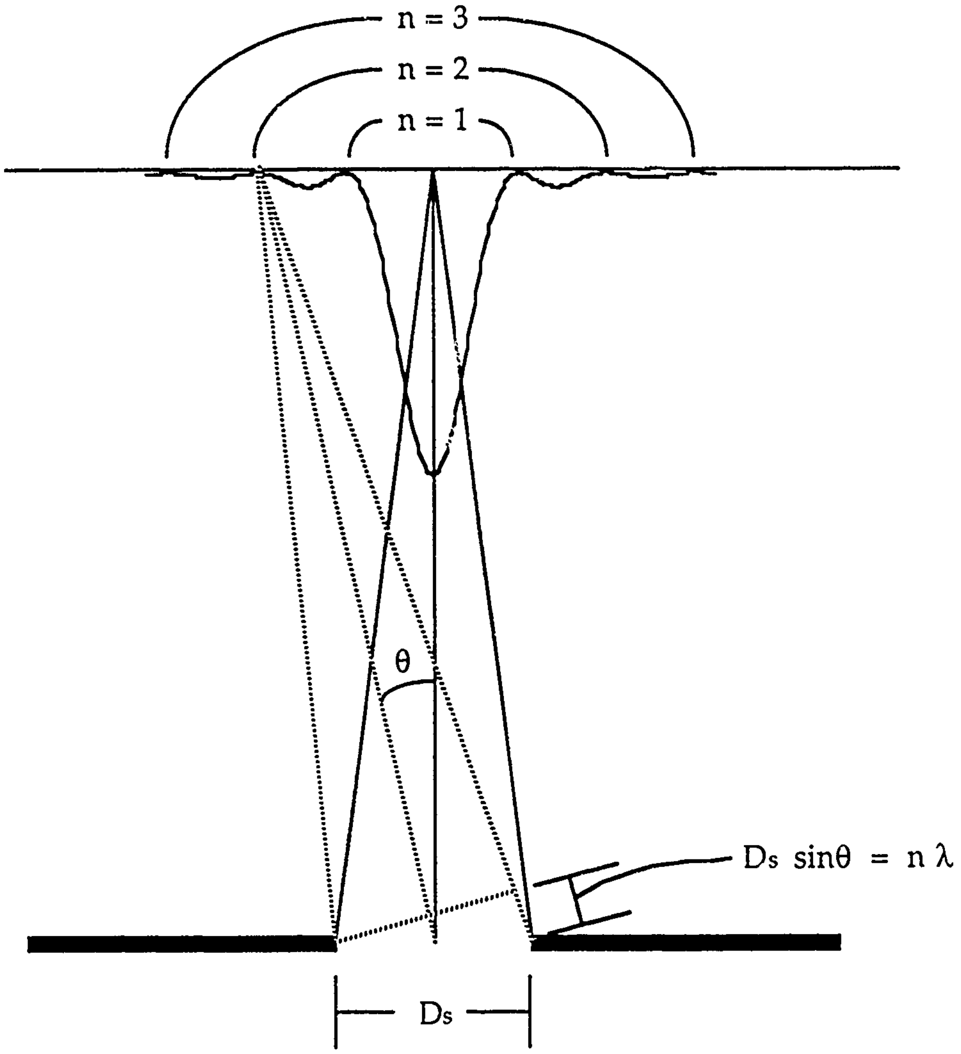


Figure B2. Minimums of the Fraunhofer Diffraction Pattern of a Slit

and the total contribution to the total amplitude A from all of the individual sources is

$$A = A(D_s, \lambda, \theta) = A(\Phi) = 2\int_{0}^{D_s/2} A_i \cos\left(\frac{x}{D_s}\Phi\right) dx$$

$$= 2\,A_i \frac{\sin\left(\frac{x}{D_s}\right)}{\frac{\Phi}{D_s}}\Bigg|_{x=0}^{x=D_s/2} = D_s\,A_i \frac{\sin\left(\frac{\Phi}{2}\right)}{\frac{\Phi}{2}} \qquad (B.6)$$

Normalizing by the amplitude at the center of the diffraction pattern gives

$$A_o = D_s A_i \lim_{\Phi \to 0} \frac{\sin\left(\frac{\Phi}{2}\right)}{\frac{\Phi}{2}} = D_s A_i \lim_{\Phi \to 0} \frac{\frac{1}{2}\cos\left(\frac{\Phi}{2}\right)}{\frac{1}{2}} = D_s A_i \qquad (B.7)$$

$$\frac{A}{A_o} = \frac{\sin\left(\frac{\Phi}{2}\right)}{\frac{\Phi}{2}} \qquad (B.8)$$

Relative intensity is more useful than relative amplitude and is given by

$$\frac{I}{I_o} = \left(\frac{A}{A_o}\right)^2 = \left(\frac{\sin\left(\frac{\Phi}{2}\right)}{\frac{\Phi}{2}}\right)^2 \qquad (B.9)$$

Figure B3 is a normalized plot of the Fraunhofer diffraction pattern using Equation B.9.

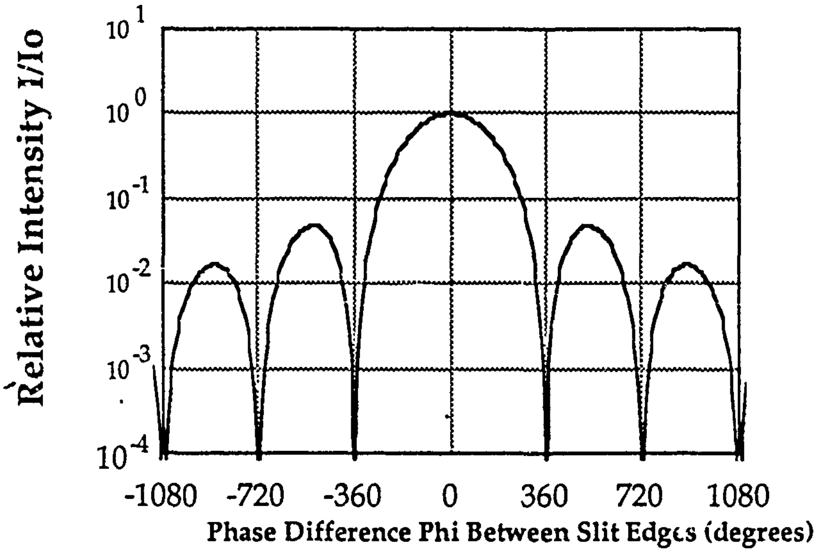


Figure B3. Normalized Fraunhofer Diffraction Pattern of a Slit

2. DIFFRACTION OF A SINGLE FIBER

The diffraction pattern of a fiber can be calculated directly using the method outlined in Kerker [Ref. 15: p. 255]. If the index of refraction of the fiber is assumed to be perfect (m = ∞), then the relative intensity of a fiber of diameter D_f can be expressed by

$$\frac{I}{I_o} = \frac{2}{K_o\, s\, \pi}\left| b_o + 2\sum_{n=1}^{\infty} b_n \cos(n\theta)\right|^2 \qquad \text{(B.10)}$$

where

$K_o = 2\pi / \lambda$

θ is the scattering angle

s is the fiber to screen distance

λ is the wavelength of the laser light

$b_n = J_n(\alpha) / H_n^{(2)}(\alpha)$

$\alpha = \pi D_f / \lambda$

$J_n(\alpha)$ are Bessel functions of the first kind

$H_n^{(2)}(\alpha)$ are Hankel functions of the second kind

[Refs. 15 and 17]

The b_n coefficients can be simplified by some simple algebra.

$$H_n^{(2)}(\alpha) = J_n(\alpha) - i\, Y_n(\alpha) \tag{B.11}$$

where $Y_n(\alpha)$ are Bessel functions of the second kind.

$$b_n = \frac{J_n(\alpha)}{H_n^{(2)}(\alpha)} = \frac{J_n(\alpha)}{J_n(\alpha) - i\ Y_n(\alpha)} \; \frac{J_n(\alpha) + i\, Y_n(\alpha)}{J_n(\alpha) + i\, Y_n(\alpha)}$$

$$= \frac{\left(J_n(\alpha)\right)^2}{\left(J_n(\alpha)\right)^2 + \left(Y_n(\alpha)\right)^2} + i\, \frac{J_n(\alpha)\, Y_n(\alpha)}{\left(J_n(\alpha)\right)^2 + \left(Y_n(\alpha)\right)^2}$$

$$= R_n + i\, S_n \tag{B.12}$$

$$R_n = \frac{\left(J_n(\alpha)\right)^2}{\left(J_n(\alpha)\right)^2 + \left(Y_n(\alpha)\right)^2} \tag{B.13}$$

$$S_n = \frac{J_n(\alpha)\, Y_n(\alpha)}{\left(J_n(\alpha)\right)^2 + \left(Y_n(\alpha)\right)^2} \tag{B.14}$$

Thus, the relative intensity equation can be rewritten

$$\frac{I}{I_o} = \frac{2}{K_o\, s\, \pi} \left[\left(R_o + 2 \sum_{n=1}^{\infty} R_n \cos(n\theta) \right)^2 + \left(S_o + 2 \sum_{n=1}^{\infty} S_n \cos(n\theta) \right)^2 \right] \tag{B.15}$$

[Ref. 14]

A. Minimum Number of Terms

Figure B4 and Figure B5 illustrate the fact that as α (which is proportional to the fiber diameter, Equation B.10) increases, the number of terms n that must be computed in each summation to achieve a specified

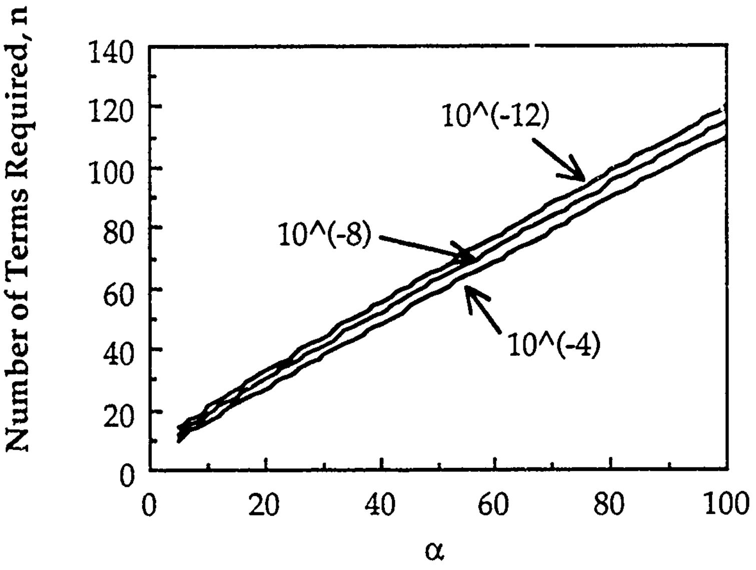


Figure B4. Summing Rn Within a Desired Accuracy

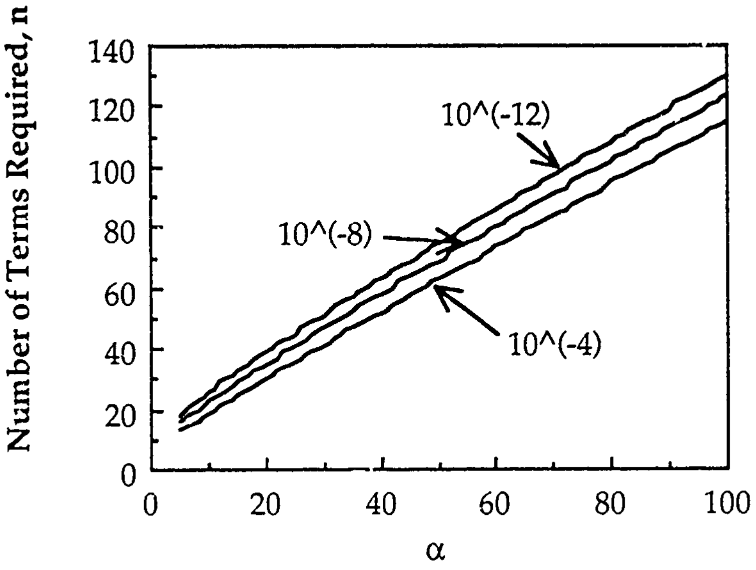


Figure B5. Summing Sn Within a Desired Accuracy

accuracy also increases. The specified accuracy can always be achieved for the R_n summation with fewer terms than for the S_n summation. The number of terms n required to achieve an accuracy of 10^{-12} for any α such that $5 < \alpha < 100$ can be predicted from a parabolic fit of the data in Figure B5.

$$n_{calculated} = 16 + 1.3\,\alpha - 0.0013\,\alpha^2 \qquad (B.16)$$

$$0 < (n_{calculated} - n_{exact}) <= 4 \qquad (B.17)$$

B. Maximum Number of Terms

For small values of α (associated with small fibers) there exists a maximum number of terms that can be used in the R_n and S_n summations. The Bessel functions of the second kind Y_n asymptotically approach $-\infty$ as α approaches zero. Underflow will occur at different values of α for different computers depending on the smallest number that can be represented. Values of α greater than five present no particular difficulty using double precision FORTRAN.

The minimum diameter fiber that can be measured with a specified laser can be calculated using the wavelength of the laser.

$$\alpha = \pi D_f / \lambda \qquad (B.18)$$

$$D_f = \alpha \lambda / \pi \qquad (B.19)$$

$$\alpha > 5 \Rightarrow (D_f)_{min} = (5 / \pi)\lambda = 1.592\ \lambda \qquad (B.20)$$

Equivalently, a laser may be selected based on the minimum diameter fiber to be measured.

$$\lambda_{min} = (\pi / 5)\, D_f = 0.628\, D_f \qquad (B.21)$$

Clearly, if very small diameter fibers are to be measured and the magnitude of the calculations on the computer to be used appears unwieldy, one potential solution would be to choose a laser with a minimum wavelength.

3. COMPARISON OF THE TWO MODELS

The Fraunhofer slit diffraction pattern and the diffraction pattern from a fiber are similar in shape, but are related in a nonlinear fashion. Figure B6 shows the diffraction patterns for a slit and a fiber with the same diameters using Equation B.9 and Equation B.10. The distance between the minimums of the slit pattern and the corresponding minimums of the fiber pattern increases progressively as the distance from the centerline increases.

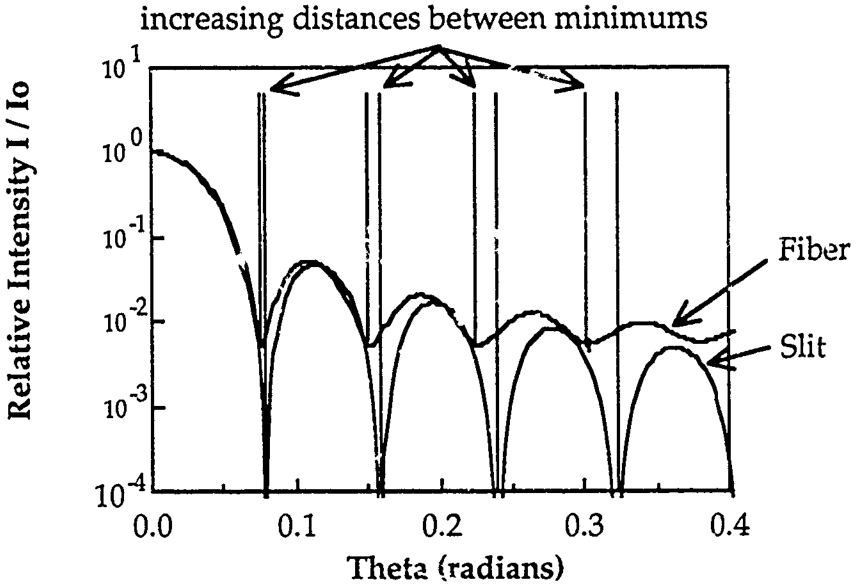


Figure B6. Diffraction Patterns of a Slit and a Fiber Having the Same Diameter

4. FIBER DIAMETER MEASUREMENT USING LASER DIFFRACTION

Diffraction pattern minimums are easily distinguished and lend themselves to measurement. Measurements of the locations of the minimums with respect to the center of the diffraction pattern are sufficient to accurately compute the diameter of the fiber or slit that caused the pattern.

One of the effects that varying D_s in Equation B.5 has is changing the location of the diffraction pattern minimums. Figure B7 shows that the diffraction pattern for a fiber with a diameter of 8 μm has its <u>first</u> minimum at the same location as the diffraction pattern for a slit with a diameter of 8.34 μm. This relationship between the slit and the fiber at the first diffraction pattern minimum can be mapped over a wide range of diameters and can be extended to the second minimum, etc. If the difference between the two diameters is expressed as a percentage of the slit diameter the relationship can be approximated very accurately by

$$(D_s - D_f) / D_s = K_1 D_s^{K_2} \qquad (B.22)$$

where K_1 and K_2 are constants and D_s and D_f are the diameters of the slit and the fiber. Figure B8 is plot of Equation B.22 for the first and sixth minimums of the diffraction patterns. Clearly, K_1 and K_2 are functions of the number of the diffraction pattern minimum as indexed from the center of the pattern.

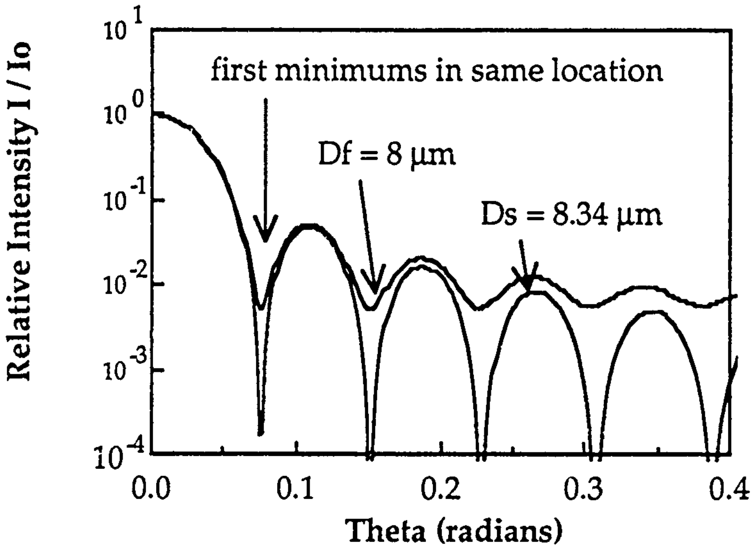


Figure B7. Fiber and Slit Having First Diffraction Pattern Minimums at the Same Location

Equation B.22 can be solved for the fiber diameter D_f

$$D_f = D_s [1 - K_1(n) D_s^{K_2(n)}] \qquad (B.23)$$

where n is the number of the diffraction pattern minimum.

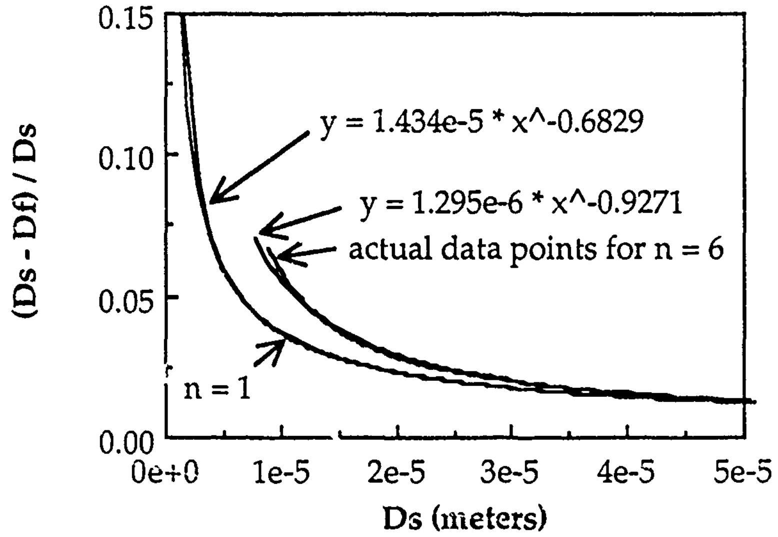


Figure B8. Percentage Error in Diameter Between a Slit and a Fiber

Note that for a given λ / Ds ratio, Equation B.4 predicts a maximum number n of observable minimums in the diffraction pattern (because $\sin\theta \leq 1$). Conversely, for a given node and specified λ there will be a limiting value of Ds such that Ds ≥ n λ. A further restriction comes from Fraunhofer diffraction theory, which makes use of a small angle approximation $\tan\theta \approx \sin\theta \approx \theta$ that limits $\theta < 15°$. For a specified λ and node number n, there exists a minimum Ds below which the relationship between slit and fiber diffraction is physically invalid. In Figure B8, with n = 6 and λ = 0.6328 µm, $(Ds)_{min}$ = (6)(0.6328 µm) / sin(15°) = 14.7 µm, although close correlation exists for Ds as small as approximately 11 µm.

Knowing the relationship between the slit diffraction pattern and the fiber diffraction pattern significantly reduces the required amount of effort required to accurately estimate the fiber diameter. The slit diffraction pattern calculations are simple, and the coefficients K_1 and K_2 need only be calculated once. Table B1 summarizes the values of K_1 and K_2 as functions of n.

Equation B.24, Equation B.25, and Equation B.26 summarize the polynomial functions $K_1(n)$ and $K_2(n)$.

TABLE B1. COEFFICIENTS FOR RELATING SLIT DIFFRACTION TO FIBER DIFFRACTION

n	K_1	K_2
1	1.434×10^{-5}	-0.6829
2	9.362×10^{-6}	-0.7247
3	5.480×10^{-6}	-0.7780
4	3.165×10^{-6}	-0.8334
5	1.890×10^{-6}	-0.8865
6	1.295×10^{-6}	-0.9271

$$(D_s - D_f) / D_s = K_1(n)\, D_s^{\,K_2(n)} \tag{B.24}$$

$$\begin{aligned} K_1(n) = {} & [\, 17.7800 - 1.01925\, n - 3.56279\, n^2 + 1.31954\, n^3 \\ & - 0.187208\, n^4 + 9.70833\text{e-}3\, n^5 \,]\, 10^{-6} \end{aligned} \tag{B.25}$$

$$\begin{aligned} K_2(n) = {} & (-0.67780) + (2.77600\text{e-}2)\, n + (-4.26917\text{e-}2)\, n^2 \\ & + (1.130\text{e-}2)\, n^3 + (-1.55833\text{e-}3)\, n^4 + (9.0\text{e-}5)\, n^5 \end{aligned} \tag{B.26}$$

where $1 \leq n \leq 6$

The most accurate results are obtained when the index n of the minimum being used is as small as possible.

5. SUMMARY

The diameter of a small fiber can be estimated to within 1 % by measuring the distance of the diffraction pattern minimums from the center of the pattern and applying Equation B.4, Equation B.23, and Equation B.24. The minimum used for measurement shouid be as close to the center of the diffraction pattern as possible. The ratio λ / D_s should be chosen such that

$$0.001 < (\lambda / D_s) < 0.5 \qquad \text{(B.27)}$$

APPENDIX C. MACINTOSH™ COMPUTER PROGRAM

This appendix contains the source code for the application CALIPER written for the Macintosh™ computer. The portions of the program that deal with the MicronEye™ were modified from Reference 18. Some subroutines were extracted essentially intact from Reference 19 and Reference 20. Programming ideas and techniques came from several of the references listed in the Bibliography. References 19 through 22 were particularly valuable programming references.

```
/* Source code for CALIPER © 1987  */
/* Written by Lieutenant Jeffrey S. Kunkel */
/* Naval Postgraduate School */

/* Portions of this program, copyright © 1984 Consulair Corp. */
/* Portions of this program written using Mac C.  Mac C is  a trademark of Consulair Corporation */

                                                        /* header files */
#include "MacDefs.h"
#include "Quickdraw.h"
#include "Window.h"
#include "Control.h"
#include "TextEdit.h"
#include "Dialog.h"
#include "Events.h"
#include "Font.h"
#include "HFSdefs.h"
#include "Memory.h"
#include "Menu.h"
#include "MiscMgr.h"
#include "OsIO.h"
#include "Packages.h"
#include "PBDefs.h"
#include "Resource.h"
#include "SANE.H"
#include "StdfileDefs.h"
#include "stdio.h"

#define   appleMRID                         1          /* apple Menu Resource ID */
#define   fileMRID                          2          /* file Menu Resource ID */
#define   quitMI                                 1     /* quit Menu Item */
#define   commandsMRID                      3          /* setup Menu Resource ID */
#define   setupMI                                1     /* setup Menu Item*/
#define   digitizeMI                             3     /* digitize Menu Item*/
#define   findNodesMI                            4     /* find nodes Menu Item*/
#define   calculateDiametersMI                   5     /* calculate Diameters Menu Item*/
```

```
#define  automaticMI                            7  /* automatic Menu Item*/
#define  scanMRID                       4          /* scan Menu Resource ID */
#define  singleSlowScanMI                       1  /* single slow scan Menu Item*/
#define  singleFastScanMI                       2  /* single fast scan Menu Item*/
#define  multipleSlowScansMI                    4  /* multiple slow scans Menu Item*/
#define  multipleFastScansMI                    5  /* multiple fast scans Menu Item*/
#define  showMicron1MI                          7  /* use Micron 1 Menu Item*/
#define  showMicron2MI                          8  /* use Micron 2 Menu Item*/

#define  controlDRID              128
#define  micronEye1RB                   1          /* MicronEye™ #1 radio button */
#define  micronEye2RB                   2          /* MicronEye™ #2 radio button */
#define  exposure1ET                    3          /* exposure #1 edit text */
#define  exposure2ET                    4          /* exposure #2 edit text */
#define  outerNode1PixelsST             12         /* outer node 1 value in pixels */
#define  innerNode1PixelsST             13         /* inner node 1 value in pixels */
#define  innerNode2PixelsST             14         /* inner node 2 value in pixels */
#define  outerNode2PixelsST             15         /* outer node 2 value in pixels */
#define  outerNode1EdgeST               17         /* outer node 1 distance to edge */
#define  innerNode1EdgeST               18         /* inner node 1 distance to edge */
#define  innerNode2EdgeST               19         /* inner node 2 distance to edge */
#define  outerNode2EdgeST               20         /* outer node 2 distance to edge */
#define  outerNode1CenterST             22         /* outer node 1 distance to center */
#define  innerNode1CenterST             23         /* inner node 1 distance to center */
#define  innerNode2CenterST             24         /* inner node 2 distance to center */
#define  outerNode2CenterST             25         /* outer node 2 distance to center */
#define  outerNode1ThetaST              27         /* outer node 1 angle */
#define  innerNode1ThetaST              28         /* inner node 1 angle */
#define  innerNode2ThetaST              29         /* inner node 2 angle */
#define  outerNode2ThetaST              30         /* outer node 2 angle */
#define  diameter1ST                    36         /* diameter using MicronEye™ #1 */
#define  diameter2ST                    37         /* diameter using MicronEye™ #2 */
#define  diameter12ST                   38         /* diameter using MicronEye™ #1 & #2 */

#define  setupDRID                129
#define  okButton                       1          /* ok button */
```

```
#define   cancelButton                       2       /* cancel button */
#define   distance_s1_ET                     8       /* distance S edit text */
#define   distance_s2_ET                     9       /* distance s edit text */
#define   distance_X1_ET                     10      /* distance X edit text */
#define   distance_X2_ET                     11      /* distance x edit text */
#define   cfsET                              21      /* CFs value edit text */
#define   cfxET                              22      /* CFx value edit text */
#define   nodeNumberET                       23      /* node number edit text */
#define   wavelengthET                       24      /* laser wavelength edit text */
#define   positiveSquelchET                  28      /* positive squelch edit text */
#define   negativeSquelchET                  29      /* negative squelch edit text */

#define   messageDRID                  131
#define   messageST                          1       /* the message text */

#define   errDRID                      9999

#define   alert1RID                    132           /* alert box for micron eye problems */
#define   alert2RID                    133           /* alert box for micron eye problems */

#define   imageWRID                    128

#define   nullSRID                     128           /* null string */
#define   micronEye1SRID               129           /* title of micron eye #1 */
#define   micronEye2SRID               130           /* title of micron eye #2 */
#define   message1SRID                 131
#define   message2SRID                 132
#define   message3SRID                 133
#define   message4SRID                 134
#define   message5SRID                 135
#define   message6SRID                 136
#define   message7SRID                 137
#define   message8SRID                 138
#define   message9SRID                 139
#define   message10SRID                140
#define   message11SRID                141
```

```
#define   message12SRID          142
#define   message13SRID          143
#define   message14SRID          144
#define   message15SRID          145
#define   message16SRID          146

#define   strtSel                0
#define   endSel                 32767
#define   SLOW                   0
#define   FAST                   1
#define   SINGLE                 0
#define   MULTIPLE               1
#define   stop10                 16384
#define   noParity               0
#define   data8                  3072
#define   baud38400              1
#define   fsCurPerm              0
#define   fsAtMark               0
#define   NIL                    0
#define   TRUE                   1
#define   FALSE                  0
#define   ON                     1
#define   OFF                    0
#define   PRINTERPORT            1
#define   MODEMPORT              0
/*****************************************************************************************/
/* Pascal to Consulair Mac C translation terms */

#define   STRING(size)      struct { unsigned char length; char text[size]; }

typedef   char              BOOLEAN;
typedef   char              CHAR;
typedef   long              LONGINT;
typedef   short             INTEGER;
typedef   unsigned char     Byte;
typedef    StringPtr              *StringHandle;

/*******************************************************************/
void        resumeProc();
void        initialize();
void        setUpMenus();
void        setUpWindows();
void        doNextEvent();
void        doMenuEvent();
void        doModalDialog();
void        initGeometry();
void        setText();
BOOLEAN     getGeometry();
BOOLEAN     getETSingle();
BOOLEAN     getETShort();
void        doModelessClick();
void        doUpdateEvent();
void        drawImageWindow();
BOOLEAN     getExposureSettings();
void        doScanNow();
void        initCamera();
void        terminateCamera();
void        displayMessage();
void        doAlertDialog1();
void        doCameraDataCorrection();
void        eraseImage();
void        doShortDisplay();
void        delayMsec();
BOOLEAN     doOneExposure();
BOOLEAN     doAlertDialog2();
BOOLEAN     bufferSizeCorrect();
void        delayTicks();
extended    timeCal();
void        initExposure();
void        fillReversedByteArray();
void        reverseByteBits();
```

```
char        *ErrMessage();
BOOLEAN     filterDigits();
BOOLEAN     doFilterDigits();
void        doFindNodes();
void        findNodes();
void        calcChipDistances();
void        calcGlobalDistances();
void        displayDistances();
void        calculateDiameters();
extended    doCalculateDiameter();
extended    coefficientK1();
extended    coefficientK2();
void        displayDiameters();
void        flipVertical();
void        autopilot();
void        digitize();

typedef struct {
    unsigned        bit0    : 1;
    unsigned        bit1    : 1;
    unsigned        bit2    : 1;
    unsigned        bit3    : 1;
    unsigned        bit4    : 1;
    unsigned        bit5    : 1;
    unsigned        bit6    : 1;
    unsigned        bit7    : 1;
} singleBits;

typedef union {
    Byte            theByte;
    singleBits      theBits;
} byteBits;

typedef struct{
    extended        value;
    Str255          text;
```

```
	BitMap		smooth;
	BitMap		raw;
	LONGINT		bufferSize;
}exposureStruct;

typedef struct{
	extended		value;
	Str255		text;
}distanceStruct;

typedef struct{
	distanceStruct	pixels;
	distanceStruct	edge;
	distanceStruct	center;
	distanceStruct	angle;
}nodeStruct;

typedef struct{
	exposureStruct	exposure;
	nodeStruct	outer;
	nodeStruct	inner;
	INTEGER		nameRID;
	ioParam		inPortIOParams;
	ioParam		outPortIOParams;
	CntrlParam	portCntrlPrms;
	BOOLEAN		show;
}micronEye;

typedef struct{
	decform		coarse;
	decform		fine;
	decform		cf;
	decform		node;
	decform		wavelength;
	decform		positiveSquelch;
	decform		negativeSquelch;
```

```
        decform         exposure;
        decform         pixels;
        decform         edge;
        decform         center;
        decform         angle;
        decform         diameter;
    }decforms;

    DialogPtr       controlDialog,messageDialog;
    WindowPtr       imageWindow;
    MenuHandle      appleMH,fileMH,commandsMH,scanMH;
    INTEGER         currentRadioButton;
    micronEye       *currentMicron,micron1,micron2;
    Str255          nullString,currentWindowTitle;
    Str255          message0,message1,message2,message3;
    Str255          message4,message5,message6,message7;
    extended        cfs,cfx,nodeNumber,wavelength;
    extended        positiveSquelch,negativeSquelch;
    extended        distance_s1,distance_s2,distance_X1,distance_X2;
    extended        diameter1,diameter2,diameterAverage;
    Str255          diameter1String,diameter2String,diameterAverageString;
    Str255          cfsString,cfxString,wavelengthString,nodeNumberString;
    Str255          positiveSquelchString,negativeSquelchString;
    Str255          distance_s1_String,distance_s2_String,distance_X1_String,distance_X2_String;
    Rect            imageContentRect;
    Byte            *reversedByteArray;
    Cursor          theCursor;
    Pattern         myBlack = {255, 255, 255, 255, 255, 255, 255, 255};
    Pattern         myWhite = {0,0,0,0,0,0,0,0};
    extended        timeConstant;
    decforms        formats;

    /********************************************************************/
    main() {
        initialize();                                   /* initialization subroutine */
        FlushEvents(everyEvent);
    InitCursor();                                   /* initialize the cursor as an arrow */

    while(TRUE) {                                   /* main loop                              */
        SystemTask();                               /* updates desk accessories that use the system clock */
        doNextEvent();                              /* do the next event */
    }
}
/***************************************** ************************/
void resumeProc() {                                 /* resume procedure if the system crashes */
    ExitToShell();                                  /* start the Finder */
}
/*******************************************************************/
void initialize() {

    DialogPtr          theDialog;

    InitFonts();                                    /* initialize the Font Manager */
    InitWindows();                                  /* initialize the Window Manager */
    InitMenus();                                    /* initialize the Menu Manager */
    TEInit();                                       /* initialize TextEdit */
    InitDialogs(resumeProc);                        /* initialize the Dialog Manager */

    initExposure(&micron1.exposure);
    initExposure(&micron2.exposure);
    setupMenus();
    setupWindows();

    theDialog = GetNewDialog(setupDRID,NIL,(WindowPtr)-1);
    getGeometry(theDialog);
    DisposDialog(theDialog);
    getExposureSettings();

    reversedByteArray = (Byte *)NewPtr(256);        /* 256 bytes */
    fillReversedByteArray(reversedByteArray);

    SetRect(&imageContentRect,0,0,512,128);
```

```
micron1.inPortIOParams.ioNamePtr    = CtoPstr(".AIn");
micron1.outPortIOParams.ioNamePtr= CtoPstr(".AOut");
micron2.inPortIOParams.ioNamePtr    = CtoPstr(".BIn");
micron2.outPortIOParams.ioNamePtr= CtoPstr(".BOut");
micron1.nameRID                = micronEye1SRID;                        /* title of micron eye 1 */
micron2.nameRID                = micronEye2SRID;                        /* title of micron eye 2 */

timeConstant = timeCal();

formats.coarse.style                 = FIXEDDECIMAL;
formats.coarse.digits                = 5;
formats.fine.style                   = FIXEDDECIMAL;
formats.fine.digits                  = 5;
formats.cf.style                     = FIXEDDECIMAL;
formats.cf.digits                    = 6;
formats.node.style                   = FIXEDDECIMAL;
formats.node.digits                  = 0;
formats.wavelength.style             = FIXEDDECIMAL;
formats.wavelength.digits            = 5;
formats.positiveSquelch.style        = FIXEDDECIMAL;
formats.positiveSquelch.digits       = 3;
formats.negativeSquelch.style        = FIXEDDECIMAL;
formats.negativeSquelch.digits       = 3;
formats.exposure.style               = FIXEDDECIMAL;
formats.exposure.digits              = 2;
formats.pixels.style                 = FIXEDDECIMAL;
formats.pixels.digits                = 0;
formats.edge.style                   = FIXEDDECIMAL;
formats.edge.digits                  = 0;
formats.center.style                 = FIXEDDECIMAL;
formats.center.digits                = 0;
formats.angle.style                  = FIXEDDECIMAL;
formats.angle.digits                 = 6;
formats.diameter.style               = FIXEDDECIMAL;
formats.diameter.digits              = 3;
```

```
}
/*********************************************************************/
void initExposure(theExposure)

     exposureStruct     *theExposure;
{
     theExposure->smooth.baseAddr   = NewPtr(8192);  /* ptr to nonrelocatable block of smoothSize bytes */
     theExposure->raw.baseAddr      = NewPtr(4096);  /* pointer to nonrelocatable block of rawSize bytes */
     theExposure->smooth.rowBytes   = 64;            /* number of columns in image window (in bytes) */
     SetRect(&theExposure->smooth.bounds,0,0,512,128);
     eraseImage(&theExposure->smooth);
}
/*********************************************************************/
void setUpMenus() {

     appleMH = GetMenu(appleMRID);                   /* get handle */
     AddResMenu(appleMH,'DRVR');                     /* Add available desk accessories */
     InsertMenu(appleMH,0);                          /* place menu after all others */

     fileMH = GetMenu(fileMRID);                     /* get handle */
     InsertMenu(fileMH,0);                           /* place menu after all others */

     commandsMH = GetMenu(commandsMRID);             /* get handle */
     InsertMenu(commandsMH,0);                       /* place menu after all others */

     scanMH = GetMenu(scanMRID);                     /* get handle */
     InsertMenu(scanMH,0);                           /* place menu after all others */
     micron1.show = TRUE;
     micron2.show = TRUE;
     CheckItem(scanMH,showMicron1MI,micron1.show);
     CheckItem(scanMH,showMicron2MI,micron2.show);

     DrawMenuBar();
}
/*********************************************************************/
void setUpWindows() {
	INTEGER         itemType;
	ControlHandle   item;
	Rect            box;

	controlDialog   = GetNewDialog(controlDRID,NIL,(WindowPtr)-1);
	imageWindow     = GetNewWindow(imageWRID,NIL,controlDialog);
	messageDialog   = GetNewDialog(messageDRID,NIL,imageWindow);

	SelIText(controlDialog,exposure1ET,strtSel,endSel);          /* cursor in first editTextItem         */
	GetDItem(controlDialog,micronEye1RB,&itemType,&item,&box);     /* get hndl to 1st image radio btn  */
	SetCtlValue(item,ON);                                         /* turn on first image radio button      */
	currentRadioButton = micronEye1RB;                            /* track current image radio button      */
	currentMicron = &micron1;

	currentWindowTitle = **(StringHandle)GetString(micronEye1SRID);  /* get image window title      */
	SetWTitle(imageWindow,&currentWindowTitle);                   /* write title on image window */


	nullString      = **(StringHandle)GetString(nullSRID);
	message0        = **(StringHandle)GetString(nullSRID);
	message1        = **(StringHandle)GetString(nullSRID);
	message2        = **(StringHandle)GetString(nullSRID);
	message3        = **(StringHandle)GetString(nullSRID);
	ParamText(&message0,&message1,&message2,&message3);    /* put null strings in param0, param1,
	                                                                   param2, param3 */
	ShowWindow(controlDialog);                                    /* make windows visible */
	ShowWindow(imageWindow);
	ShowWindow(messageDialog);
}
/*****************************************************************/
void doNextEvent() {

	EventRecord   theEvent;
	CHAR          keyHit;
	short         mouseLocation;
```

```
WindowPtr     whichWindow;
DialogPtr     theDialog;
INTEGER       eventItemHit;

GetNextEvent(everyEvent, &theEvent);                    /* get the next event and store in theEvent */
if(IsDialogEvent(&theEvent)) {                                    /* is it a modeless dialog event? */
    if((theEvent.what == keyDown) && (theEvent.modifiers & cmdKey)) {   /* cmd key ? */
        keyHit = theEvent.message & charCodeMask;       /* keyboard character */
        doMenuEvent(MenuKey(keyHit));
        HiliteMenu(0);                                            /* unhighlights the menu */
    } else if (DialogSelect(&theEvent,&theDialog,&eventItemHit)) {
        doModelessClick(eventItemHit);                  /* handle enabled dialog item event */
    }
} else {                                                /* else handle non dialog event */
    switch(theEvent.what) {                                       /* branch to appropriate event */
        case keyDown:                                             /* keydown event */
            keyHit = theEvent.message & charCodeMask;             /* keyboard character  */
            if(theEvent.modifiers & cmdKey) {           /* cmd key short cut requested? */
                doMenuEvent(MenuKey(keyHit));
            HiliteMenu(0);                              /* unhighlights the menu */
            }
            break;                                      /* done with the event */
        case mouseDown:                                 /* mouse clicked */
            mouseLocation = FindWindow(&theEvent.where,&whichWindow);
            switch(mouseLocation) {                     /* branch to mouse location */
                case inMenuBar:                         /* mouse in the menu bar */
                    doMenuEvent(MenuSelect(&theEvent.where));
                    break;
                case inSysWindow:                       /* mouse in a system window */
                    SystemClick(&theEvent,whichWindow); /* mouse click in desk accessory */
                    break;
                case inContent:                         /* mouse in the content region of a window */
                    if(whichWindow == imageWindow) break;  /* skip image window */
                    if(whichWindow == messageDialog) break;  /* skip msg dialog window */
                    SelectWindow(whichWindow);             /* whichWindow active window */
                    break;
                    default:
                         break;
               }
               break;                                                    /* done with the event */
          case updateEvt:
               doUpdateEvent(&theEvent);
               break;
          default:
               break;
     }
  }
}
/*******************************************************************/
void doMenuEvent(menuCode)
     LONGINT        menuCode;
{
     INTEGER        menuTitle,menuItem,refNum;
     DialogPtr      theDialog;
     GrafPtr        theCurrentPort;
     Str255         deskAccessoryName;

     menuTitle      = HiWord(menuCode);                  /* identify which menu */
     menuItem       = LoWord(menuCode);                  /* identify the specific menu item */

     switch(menuTitle) {                                 /* branch to the appropriate menu */
          case appleMRID:                                /* hndl an event in the desk accessory menu */
               GetItem(appleMH,menuItem,&deskAccessoryName);
          GetPort(&theCurrentPort);                      /* record the current graf port */
               refNum = OpenDeskAcc(&deskAccessoryName);  /* open the desk accessory */
               SetPort(theCurrentPort);                  /* ensure the original graf port is restored */
               break;
          case fileMRID:                                 /* handle an event in the file menu */
               switch(menuItem) {
                    case quitMI:
                         ExitToShell();                  /* return to the Finder */
                         break;
```

```
				default:
					break;
			}
			break;
		case commandsMRID:
			switch(menuItem) {
				case setupMI:
					theDialog = GetNewDialog(setupDRID,NIL,(WindowPtr)-1);
					doModalDialog(theDialog);			/* handle the modal dialog */
					DisposDialog(theDialog);			/* closes the dialog and releases the memory */
					break;
				case digitizeMI:
					digitize();
					break;
				case findNodesMI:
					findNodes();
					break;
				case calculateDiametersMI:
					calculateDiameters();
					break;
				case automaticMI:
					autopilot();
					break;
				default:
					break;
			}
			break;
		case scanMRID:
			switch(menuItem) {
				case singleSlowScanMI:
					doScanNow(SLOW,SINGLE);
					break;
				case singleFastScanMI:
					doScanNow(FAST,SINGLE);
					break;
				case multipleSlowScansMI:
```

```
                                        doScanNow(SLOW,MULTIPLE);
                                        break;
                                case multipleFastScansMI:
                                        doScanNow(FAST,MULTIPLE);
                                        break;
                                case showMicron1MI:
                                        micron1.show = !micron1.show;
                                        CheckItem(scanMH,showMicron1MI,micron1.show);
                                        break;
                                case showMicron2MI:
                                        micron2.show = !micron2.show;
                                        CheckItem(scanMH,showMicron2MI,micron2.show);
                                        break;
                                default:
                                        break;
                        }
                        break;
                default:
                        break;
        }
        HiliteMenu(0);
}
/*****************************************************************/
void doModalDialog(theDialog)                         /* handle the settings dialog box */

        DialogPtr       theDialog;
{
        INTEGER         itemHit,modalFlag,theStage;

        initGeometry(theDialog);                      /* initialize and record the check box settings */
        ShowWindow(theDialog);                        /* display the dialog box */

        modalFlag = ON;                               /* flag to exit dialog routine */

        while (modalFlag == ON) {                     /* until ok or cancel buttons are hit */
                ModalDialog((ProcPtr)NIL,&itemHit);   /* handles events in the dialog window */
        switch(itemHit) {                               /* branch to the appropriate dialog event */
            case okButton:                              /* ok button hit */
                if (getGeometry(theDialog)) {           /* get system geometry */
                    modalFlag = OFF;                    /* turn flag off */
                } else {
                    theStage = GetAlrtStage();
                    switch (theStage) {
                            case 0:
                            doAlertDialog1(message10SRID);
                                    break;
                            case 1:
                            doAlertDialog1(message11SRID);
                                    break;
                            case 2:
                            doAlertDialog1(message12SRID);
                                    break;
                            case 3:
                            doAlertDialog1(message13SRID);
                                    break;
                    }
                }
                break;
            case cancelButton:                          /* cancel button hit */
                modalFlag = OFF;                        /* turn flag off */
                break;
        }
    }
    ResetAlrtStage();
}
/***** ***************************************************************************/
void initGeometry(theDialog)

    DialogPtr           theDialog;
{
    setText(theDialog,distance_s1_ET,&distance_s1_String);
    setText(theDialog,distance_s2_ET,&distance_s2_String);
```

```
	setText(theDialog,distance_X1_ET,&distance_X1_String);
	setText(theDialog,distance_X2_ET,&distance_X2_String);
	setText(theDialog,cfsET,&cfsString);
	setText(theDialog,cfxET,&cfxString);
	setText(theDialog,nodeNumberET,&nodeNumberString);
	setText(theDialog,wavelengthET,&wavelengthString);
	setText(theDialog,positiveSquelchET,&positiveSquelchString);
	setText(theDialog,negativeSquelchET,&negativeSquelchString);
}
/*******************************************************************/
void setText(theDialog,theEditTextItem,theString)

	DialogPtr		theDialog;
	INTEGER		theEditTextItem;
	Str255		*theString;
{
	ControlHandle	itemHandle;
	INTEGER		itemType;
	Rect			box;

	GetDItem(theDialog,theEditTextItem,&itemType,&itemHandle,&box);		/* get a handle */
	SetIText(itemHandle,theString);					/* set the text in the edit text item */
}
/*******************************************************************/
BOOLEAN getGeometry(theDialog)

	DialogPtr		theDialog;
{
	if (!getETSingle(theDialog,distance_s1_ET,&distance_s1,&distance_s1_String))		return(FALSE);
	if (!getETSingle(theDialog,distance_s2_ET,&distance_s2,&distance_s2_String))		return(FALSE);
	if (!getETSingle(theDialog,distance_X1_ET,&distance_X1,&distance_X1_String))		return(FALSE);
	if (!getETSingle(theDialog,distance_X2_ET,&distance_X2,&distance_X2_String))		return(FALSE);
	if (!getETSingle(theDialog,cfsET,&cfs,&cfsString))						return(FALSE);
	if (!getETSingle(theDialog,cfxET,&cfx,&cfxString))						return(FALSE);
	if (!getETSingle(theDialog,nodeNumberET,&nodeNumber,&nodeNumberString))		return(FALSE);
	if (!getETSingle(theDialog,wavelengthET,&wavelength,&wavelengthString))		return(FALSE);
	if (!getETSingle(theDialog,positiveSquelchET,&positiveSquelch,&positiveSquelchString))	return(FALSE);
	if (!getETSingle(theDialog,negativeSquelchET,&negativeSquelch,&negativeSquelchString))
	return(FALSE);
}
/*****************************************************************/
BOOLEAN getETSingle(theDialog,theEditTextItem,theVariable,theString)

	DialogPtr		theDialog;
	INTEGER		theEditTextItem;
	extended		*theVariable;
	Str255		*theString;
{
	ControlHandle	itemHandle;
	INTEGER		itemType;
	Rect			box;
	Str255		theText;
	BOOLEAN		procede;
	extended		test;

	GetDItem(theDialog,theEditTextItem,&itemType,&itemHandle,&box);
	GetIText(itemHandle,&theText);
	PtoCstr(&theText);
	test = atof(&theText);
	CtoPstr(&theText);
	if ((test >= 0.0) && (test <= 1000.0) && filterDigits(&theText)){
		*theVariable = test;
		*theString = theText;
		return(TRUE);
	} else {
		return(FALSE);
	}
}
/*****************************************************************/
void doModelessClick(itemHit)

	INTEGER		itemHit;
```

```
{
    ControlHandle       imageRBH;
    INTEGER             itemType;
    Rect                box;
    GrafPtr             theCurrentPort;

    if (itemHit != currentRadioButton) {
        if (itemHit == micronEye1RB) {
            GetDItem(controlDialog,micronEye2RB,&itemType,&imageRBH,&box);
            SetCtlValue(imageRBH,OFF);                  /* turn off image 2 radio button */
            GetDItem(controlDialog,micronEye1RB,&itemType,&imageRBH,&box);
            SetCtlValue(imageRBH,ON);                   /* turn on image 1 radio button */
            currentRadioButton = micronEye1RB;          /* record current radio button */
            currentMicron = &micron1;
            currentWindowTitle = **(StringHandle)GetString(micronEye1SRID);
            SetWTitle(imageWindow,&currentWindowTitle);
        } else {
            GetDItem(controlDialog,micronEye1RB,&itemType,&imageRBH,&box);
            SetCtlValue(imageRBH,OFF);
            GetDItem(controlDialog,micronEye2RB,&itemType,&imageRBH,&box);
            SetCtlValue(imageRBH,ON);
            currentRadioButton = micronEye2RB;
            currentMicron = &micron2;
            currentWindowTitle = **(StringHandle)GetString(micronEye2SRID);
            SetWTitle(imageWindow,&currentWindowTitle);
        }
        GetPort(&theCurrentPort);
        SetPort(imageWindow);
        InvalRect(&imageContentRect);                   /* generate an update event */
        SetPort(theCurrentPort);
    }
}
/******************************************************************/
void doUpdateEvent(theEvent)

    EventRecord    *theEvent;
```

```
{
	WindowPtr	updateWindow;
	GrafPtr		theCurrentPort;

	GetPort(&theCurrentPort);				/* save current port */
	updateWindow = (WindowPtr)theEvent->message;		/* get WindowPtr to be updated */
	SetPort(updateWindow);					/* set as current port */
	BeginUpdate(updateWindow);				/* begin update process */
		if(updateWindow == messageDialog) DrawDialog(messageDialog);
		if(updateWindow == imageWindow) drawImageWindow(currentMicron);
	EndUpdate(updateWindow);
	SetPort(theCurrentPort);				/* restore current port */
}
/****************************************************************/
void drawImageWindow(theMicron)

	micronEye		*theMicron;
{
	Rect			rect1,rect2,rect3,rect4;

	SetRect(&rect1,0,0,512,32);
	SetRect(&rect2,0,32,512,64);
	SetRect(&rect3,0,64,512,96);
	SetRect(&rect4,0,96,512,128);
	CopyBits(&theMicron->exposure.smooth,&imageWindow->portBits,&rect1,&rect1,srcCopy,NIL);
	CopyBits(&theMicron->exposure.smooth,&imageWindow->portBits,&rect2,&rect2,srcCopy,NIL);
	CopyBits(&theMicron->exposure.smooth,&imageWindow->portBits,&rect3,&rect3,srcCopy,NIL);
	CopyBits(&theMicron->exposure.smooth,&imageWindow->portBits,&rect4,&rect4,srcCopy,NIL);
}
/****************************************************************/
BOOLEAN getExposureSettings() {

	if (!getETSingle(controlDialog,exposure1ET,&micron1.exposure.value,&micron1.exposure.text))
				return(FALSE);
	if (!getETSingle(controlDialog,exposure2ET,&micron2.exposure.value,&micron2.exposure.text))
				return(FALSE);
```

```
     return(TRUE);
}
/**** ***********************************************************/
void doScanNow(theMode,theScanType)

     BOOLEAN      theMode;                          /* FAST or SLOW */
     BOOLEAN      theScanType;                      /* SINGLE or MULTIPLE */
{
     INTEGER      eventMask,theStage;
     BOOLEAN      procede;
     GrafPtr      theCurrentPort;
     EventRecord  theEvent;

     theCursor = **GetCursor(watchCursor);
     SetCursor(&theCursor);
     displayMessage(message1SRID,nullSRID,nullSRID,nullSRID);

     if (!getExposureSettings()) {
          theStage = GetAlrtStage();
          switch (theStage) {
               case 0:
                    doAlertDialog1(message6SRID);
                    break;
               case 1:
                    doAlertDialog1(message7SRID);
                    break;
               case 2:
                    doAlertDialog1(message8SRID);
                    break;
               case 3:
                    doAlertDialog1(message9SRID);
                    break;
          }
          displayMessage(nullSRID,nullSRID,nullSRID,nullSRID);
          InitCursor(),
          return;
}
ResetAlrtStage();

eventMask = mDownMask + keyDownMask;

do {
      initCamera(&micron1);
      initCamera(&micron2);

      if(!EventAvail(eventMask,&theEvent)) {
            procede = doOneExposure(&micron1,theMode);
      } else {
            procede = FALSE;
      }
      if(procede && !EventAvail(eventMask,&theEvent)) procede = doOneExposure(&micron2,theMode);

      terminateCamera(&micron1);
      terminateCamera(&micron2);

      ResetAlrtStage();

      if (procede) {
            if(!EventAvail(eventMask,&theEvent))
                  doCameraDataCorrection(&micron1.exposure,theMode);
            if(!EventAvail(eventMask,&theEvent))
                  doCameraDataCorrection(&micron2.exposure,theMode);
      }

      displayMessage(nullSRID,nullSRID,nullSRID,nullSRID);

      if (micron1.show && micron2.show) {
            if(!EventAvail(eventMask,&theEvent)) doShortDisplay(&micron1);
            if(!EventAvail(eventMask,&theEvent)) doShortDisplay(&micron2);
      } else drawImageWindow(currentMicron);

} while (theScanType && !EventAvail(eventMask,&theEvent) && procede);
```

```
	currentWindowTitle = **(StringHandle)GetString(currentMicron->nameRID);
	SetWTitle(imageWindow,&currentWindowTitle);

	GetPort(&theCurrentPort);
	SetPort(imageWindow);
	InvalRect(&imageContentRect);		/* generate an update event for the new image window */
	SetPort(theCurrentPort);

	InitCursor();
}
/******************************************************************************/
void initCamera(theCamera)					/* initialize one camera and its port */

	micronEye	*theCamera;
{
	INTEGER	configuration;
	OSErr		error;

	displayMessage(message2SRID,theCamera->nameRID,nullSRID,nullSRID);

	configuration = stop10 + noParity + data8 + baud38400;		/* 1 stop bit, no parity, 8 data bits, 38400 baud */

	theCamera->inPortIOParams.ioCompletion			= NIL;	/* completion routine */
	theCamera->inPortIOParams.ioPermssn= fsCurPerm;
	error = PBOpen(&theCamera->inPortIOParams,TRUE);
	theCamera->outPortIOParams.ioCompletion			= NIL;
	theCamera->outPortIOParams.ioPermssn= fsCurPerm;
	error = error + PBOpen(&theCamera->outPortIOParams,TRUE);

	theCamera->portCntrlPrms.ioCompletion			= NIL;
	theCamera->portCntrlPrms.ioRefNum			= theCamera->inPortIOParams.ioRefNum;
	theCamera->portCntrlPrms.CSCode			= 8;		/* ctl code for configuration information */
	theCamera->portCntrlPrms.csp.asncConfig			= configuration;
	error = error + PBControl(&theCamera->portCntrlPrms,TRUE);	/* send the ctl code & info to the port */
	theCamera->portCntrlPrms.ioRefNum= theCamera->outPortIOParams.ioRefNum;
	error = error + PBControl(&theCamera->portCntrlPrms,TRUE);
	if (error != noErr) OSError("doInitCamera",error,"trouble initializing input and/or output port(s)");
}
/****************************************************************/
void terminateCamera(theCamera)

	micronEye	*theCamera;
{
	OSErr		error;

	error = PBClose(&theCamera->inPortIOParams,TRUE);
	if (error != noErr) OSError("PBClose",error,"closing the input port");
}
/****************************************************************/
void displayMessage(message0ID,message1ID,message2ID,message3ID)

	INTEGER		message0ID,message1ID,message2ID,message3ID;
{
	Str255		theMessage;

	theMessage = **(StringHandle)GetString(message0ID);
	strcpy(&message0,&theMessage);
	theMessage = **(StringHandle)GetString(message1ID);
	strcpy(&message1,&theMessage);
	theMessage = **(StringHandle)GetString(message2ID);
	strcpy(&message2,&theMessage);
	theMessage = **(StringHandle)GetString(message3ID);
	strcpy(&message3,&theMessage);

	ParamText(&message0,&message1,&message2,&message3);
	DrawDialog(messageDialog);
}
/****************************************************************/
void doAlertDialog1(theSRID)
```

```
    INTEGER          theSRID;
{
    message4 = **(StringHandle)GetString(theSRID);
    ParamText(&message4,&nullString,&nullString,&nullString);
    Alert(alert1RID,NIL);
    ParamText(&message0,&message1,&message2,&message3);
}
/*******************************************************************/
void doCameraDataCorrection(theMicron,theMode)

    micronEye            *theMicron;
    BOOLEAN              theMode;
{
    exposureStruct       *theExposure;
    register INTEGER     row,col,x,y,i;
    LONGINT              bitNum;
    Byte                 *bytePtr;

    if (theMicron->show == FALSE) return;

    displayMessage(message5SRID,theMicron->nameRID,nullSRID,nullSRID);

    theExposure = &theMicron->exposure;

    bytePtr = (Byte *)theExposure->raw.baseAddr;
    for (i = 0; i < theExposure->bufferSize; i++) {
        *(bytePtr + i) = reversedByteArray[*(bytePtr + i)];
    }

    switch(theMode) {
        case FAST:
            CopyBits (&theExposure->raw,&theExposure->smooth,
                &theExposure->raw.bounds,&theExposure->smooth.bounds,srcCopy,NIL);
            break;
        case SLOW:
            eraseImage(&theExposure->smooth);        /* set all bits to one (black) */
            for (row = 0; row < 128; row++) {
                for (col = 0; col < 256; col++) {
                    bitNum = row * 256 + col;
                    if (BitTst(theExposure->raw.baseAddr,bitNum)) {
                        if (row % 2 != 0) {
                                if (col % 2 != 0) {
                                y = row + 1;
                                x = 2 * col;
                            } else {
                                y = row;
                                x = 2 * col + 3;
                            }
                        } else {
                                if (col % 2 != 0) {
                                y = row;
                                x = 2 * col + 3;
                            } else {
                                y = row + 1;
                                x = 2 * col;
                            }
                        }
                        if ((x > 0) && (x < 513) && (y < 128)) {
                            bitNum = x - 1 + 512 * y;
                            BitSet(theExposure->smooth.baseAddr,bitNum);
                        }
                    }
                }
            }
            break;
    }
    flipVertical(&theExposure->smooth);
}
/**********************************************************************/
void eraseImage(theBitMapPtr)
```

```
    BitMap          *theBitMapPtr;
{
    GrafPtr         theCurrentPort,theTempPort;

    GetPort(&theCurrentPort);
    theTempPort = (GrafPtr)NewPtr(sizeof(GrafPort));
    OpenPort(theTempPort);
    SetPortBits(theBitMapPtr);
    FillRect(&theBitMapPtr->bounds,&myWhite);
    SetPort(theCurrentPort);
    ClosePort(theTempPort);
    DisposPtr((Ptr)theTempPort);
}
/**********************************************************************/
void doShortDisplay(theMicron)

    micronEye       *theMicron;
{
    extended        theDelay;

    if (theMicron->show == FALSE) return;
    theDelay = 1000;

    currentWindowTitle = **(StringHandle)GetString(theMicron->nameRID);
    SetWTitle(imageWindow,&currentWindowTitle);
    drawImageWindow(theMicron);

    delayMsec(&theDelay,&timeConstant);
}
/**********************************************************************/
void delayMsec(theDelay,theTimeConstant)

    extended        *theTimeConstant,*theDelay;
{
    LONGINT         i;
    extended        doneCount;
```

```
        doneCount = *theDelay * *theTimeConstant;
        for (i = 0; i <= doneCount;i++) {
            /* do nothing */
        }
}
/********************************************************************/
BOOLEAN doOneExposure(theMicron,theMode)

        micronEye           *theMicron;
        BOOLEAN             theMode;
{
        INTEGER             error, count, i,maxCount,dataOK;
        CHAR                cameraCommand1[1],cameraCommand2[1],cameraCommand3[1];
        LONGINT             commandByteLength;
        exposureStruct      *theExposure;
        extended            theDelay;

        if (theMicron->show == FALSE) return(TRUE);

        displayMessage(message3SRID,theMicron->nameRID,nullSRID,nullSRID);

        theExposure = &theMicron->exposure;
        theDelay = theExposure->value * 1000;

        theMicron->portCntrlPrms.ioRefNum= theMicron->inPortIOParams.ioRefNum;
        theMicron->portCntrlPrms.CSCode                   = 9;    /* control code for asynchronous info */
        theMicron->portCntrlPrms.csp.asyncInBuff.asncBPtr = theExposure->raw.baseAddr;
        theMicron->portCntrlPrms.csp.asyncInBuff.asncBLen = 4097;                /* max buffer length */
        error = PBControl(&theMicron->portCntrlPrms,TRUE);
        switch(theMode) {
            case FAST:
                cameraCommand1[0] =     0xD8;              /* initialize OpticRam and read data */
                cameraCommand2[0] =     0xD9;              /* initialize OpticRam and read data */
                cameraCommand3[0] =     0xDB;              /* freeze OpticRam without transmit */
                theExposure->bufferSize = 1024;            /* 128 x 64 / 8 */
            theExposure->raw.rowBytes = 16;
            SetRect(&theExposure->raw.bounds,0,0,128,64);
            break;
        case SLOW:
            cameraCommand1[0] =     0xF8;           /* initialize OpticRam and read data */
            cameraCommand2[0] =     0xF9;           /* initialize OpticRam and read data */
            cameraCommand3[0] =     0xFB;           /* freeze OpticRam without transmit */
            theExposure->bufferSize = 4096;         /* 256 x 128 / 8 */
            theExposure->raw.rowBytes = 32;
            SetRect(&theExposure->raw.bounds,0,0,256,128);
            break;
    }

    commandByteLength = 1;
    maxCount = 3;

    theMicron->outPortIOParams.ioBuffer= cameraCommand1;        /* command for the micron Eye */
    theMicron->outPortIOParams.ioReqCount         = commandByteLength;
    theMicron->outPortIOParams.ioPosMode= fsAtMark;             /* write at current position */

    i = 0;
    do {
        i++;
        count = 0;
        do {
            ++count;
            error = error + PBKillIO(&theMicron->inPortIOParams,TRUE);
            error = error + PBKillIO(&theMicron->outPortIOParams,TRUE);
            error = error + PBWrite(&theMicron->outPortIOParams,TRUE);
            if (error != noErr) OSError("doOneExposure",error,"first loop");
        } while ((!bufferSizeCorrect(theMicron)) && (count < maxCount));
        if (count >= maxCount) return(doAlertDialog2(theMicron));
        theMicron->outPortIOParams.ioBuffer         = cameraCommand2;
        error = error + PBWrite(&theMicron->outPortIOParams,TRUE);
        error = error + PBKillIO(&theMicron->inPortIOParams,TRUE);
        error = error + PBKillIO(&theMicron->outPortIOParams,TRUE);
```

```
		theMicron->outPortIOParams.ioBuffer		= cameraCommand3;
		error = error + PBWrite(&theMicron->outPortIOParams,TRUE);
		delayMsec(&theDelay,&timeConstant);
		theMicron->outPortIOParams.ioBuffer		= cameraCommand1;
		error = error + PBWrite(&theMicron->outPortIOParams,TRUE);
		if (error != noErr) OSError("doOneExposure",error,"main loop");

	} while ((dataOK = !bufferSizeCorrect(theMicron)) && (i < maxCount));
	if (i >= maxCount) return(doAlertDialog2(theMicron));

	return(TRUE);
}
/*********************************************************************/
BOOLEAN doAlertDialog2(theMicron)

	micronEye	*theMicron;
{

	INTEGER		itemHit;
	Str255		alertMessage0,alertMessage1;

	message4 = **(StringHandle)GetString(theMicron->nameRID);
	message5 = **(StringHandle)GetString(message4SRID);
	ParamText(&message4,&message5,&nullString,&nullString);
	itemHit = Alert(alert2RID,NIL);
	ParamText(&message0,&message1,&message2,&message3);
	return(itemHit - 1);
}
/*********************************************************************/
BOOLEAN bufferSizeCorrect(theMicron)

	micronEye		*theMicron;
{
	LONGINT			lastBufferSize;
	INTEGER			error;
```

```
    exposureStruct        *theExposure;

    theMicron->portCntrlPrms.CSCode = 2;
    theMicron->portCntrlPrms.ioRefNum = theMicron->inPortIOParams.ioRefNum;
    error = PBStatus(&theMicron->portCntrlPrms,TRUE);
    if (error != noErr) OSError("bufferSizeCorrect",error,"trouble interrogating # of buffered bytes");

    do {
        lastBufferSize = theMicron->portCntrlPrms.csp.asyncNBytes;
        delayTicks(5);
                /* delay in ticks */
        error = PBStatus(&theMicron->portCntrlPrms,TRUE);
        if (error != noErr) OSError("bufferSizeCorrect",error,"first loop");
    } while (lastBufferSize != theMicron->portCntrlPrms.csp.asyncNBytes);

    return(lastBufferSize == theMicron->exposure.bufferSize);
}
/***********************************************************************/
void delayTicks(ticks)

    LONGINT       ticks;
{
    LONGINT       finish;

    finish = TickCount() + ticks;
    while (finish > TickCount()) {
        /* do nothing */
    }
}
/***********************************************************************/
extended timeCal() {                /* determines the number of loops required fr  1 msec */

    LONGINT             startTicks,tickTime;
    extended  theTimeConstant,theDelay,targetTickTime;
    theDelay = 1000.0;
    targetTickTime = theDelay * 60.0 / 1000.0;
    theTimeConstant = 1.300;                                    /* initial guess */
    do {
        startTicks = TickCount();
        delayMsec(&theDelay,&theTimeConstant);
        tickTime = TickCount() - startTicks;
        theTimeConstant *= targetTickTime/tickTime;
    } while(fabs(targetTickTime - tickTime) > 1.0);
    return(theTimeConstant);
}
/**********************************************************************/
void fillReversedByteArray(theArray)

    Byte            theArray[];
{
    INTEGER     i;

    for (i = 0; i < 256; i++) {
        *(theArray + i) = i;
        reverseByteBits(theArray + i);
    }
}

/**********************************************************************/
void reverseByteBits(theBytePtr)                    /* in place reversal of bits in one byte */

    Ptr             theBytePtr;
{
    byteBits        byteCopy,*byteOrig;

    byteCopy.theByte = *theBytePtr;
    byteOrig = (byteBits *)theBytePtr;

    byteOrig->theBits.bit0 = !byteCopy.theBits.bit7;
    byteOrig->theBits.bit1 = !byteCopy.theBits.bit6;
    byteOrig->theBits.bit2 = !byteCopy.theBits.bit5;
    byteOrig->theBits.bit3 = !byteCopy.theBits.bit4;
    byteOrig->theBits.bit4 = !byteCopy.theBits.bit3;
    byteOrig->theBits.bit5 = !byteCopy.theBits.bit2;
    byteOrig->theBits.bit6 = !byteCopy.theBits.bit1;
    byteOrig->theBits.bit7 = !byteCopy.theBits.bit0;
}
/*****************************************************************/
BOOLEAN filterDigits(theText)

    Str255          *theText;
{
    Str255          textCopy;

    PtoCstr(theText);
    strcpy(&textCopy,theText);
    CtoPstr(theText);

    return(doFilterDigits(&textCopy));
}
/*****************************************************************/
BOOLEAN doFilterDigits(theText)

    CHAR            theText[];
{
    INTEGER         i;

    for (i = 0; i <= strlen(theText); i++) {
        if (((theText[i] < '0') || (theText[i] > '9')) && (theText[i] != '\0') && (theText[i] != '.')) {
            return(FALSE);
        }
    }
    return(TRUE);
}
/*****************************************************************/
void findNodes() {

    doFindNodes(&micron1,1);
```

```
	doFindNodes(&micron2,2);
	displayMessage(message15SRID,nullSRID,nullSRID,nullSRID);
	calcChipDistances(&micron1,1);
	calcChipDistances(&micron2,2);
	calcGlobalDistances();
	displayDistances(&micron1);
	displayDistances(&micron2);
	displayMessage(nullSRID,nullSRID,nullSRID,nullSRID);
}
/***** *****************************************************************/
void doFindNodes(theMicron,ID)

	micronEye		*theMicron;
	INTEGER			ID;
{
	register INTEGER	col,row;
	INTEGER			sum[514];
	INTEGER			theMaxCol,theMaxVal,theMinVal;
	INTEGER			posFlag,negFlag,posLevel,negLevel;
	INTEGER			posNode,negNode;
	LONGINT			bitNum;
	exposureStruct		*theExposure;
	decimal			d;
	char			s;

	displayMessage(message14SRID,theMicron->nameRID,nullSRID,nullSRID);

	theExposure = &theMicron->exposure;

	for(col = 0; col < 514; col++) {
		sum[col] = 0;
	}

	for(col = 2; col < 512; col++) {
		for(row = 2; row < 129; row++) {
			bitNum = 512 * row + col;
```

```
        if (BitTst(theExposure->smooth.baseAddr,bitNum)) {
            sum[col-2]    += 1;
            sum[col-1]    += 1;
            sum[col]      += 1;
            sum[col+1]    += 1;
            sum[col+2]    += 1;
        }
    }
}

theMaxCol = 0;
theMaxVal = 0;
theMinVal = 3000,
for (col = 2; col < 512; col++) {
    if (sum[col] < theMinVal) {
        theMinVal = sum[col];
    }
    if (sum[col] > theMaxVal) {
        theMaxVal = sum[col];
        theMaxCol = col;
    }
}
col = 0;
posFlag = OFF;
negFlag = OFF;
posLevel = theMaxVal - (theMaxVal - theMinVal) * positiveSquelch;
negLevel = theMaxVal - (theMaxVal - theMinVal) * negativeSquelch;
while (col < 512) {
    col++;
    if (posFlag == OFF) {
        if (sum[col] > posLevel) {
            posNode = col;
            posFlag = ON;
        }
    } else if (negFlag == OFF) {
        if (sum[col] < negLevel) {
                negNode = col;
                negFlag = ON;
            }
        } else break;
    }

    if (ID == 1) {
        theMicron->outer.pixels.value = posNode;
        theMicron->inner.pixels.value = negNode;
    } else {
        theMicron->outer.pixels.value = negNode;
        theMicron->inner.pixels.value = posNode;
    }

    ext2Str(&formats.pixels,theMicron->outer.pixels.value,&theMicron->outer.pixels.text);
    ext2Str(&formats.pixels,theMicron->inner.pixels.value,&theMicron->inner.pixels.text);
}
/***********************************************************************/
void calcChipDistances(theMicron,ID)

    micronEye     *theMicron;
    INTEGER       ID;
{
    extended  edgeReference;
    extended  perPixel;

    perPixel = 4420.0 / 514.0;                          /* µm per pixel */

    if (ID == 1) {
        edgeReference = 514.0;
    } else {
        edgeReference = 0.0;
    }

    theMicron->outer.edge.value = fabs((edgeReference - theMicron->outer.pixels.value) * perPixel);
    theMicron->inner.edge.value = fabs((edgeReference - theMicron->inner.pixels.value) * perPixel);
```

```
	ext2Str(&formats.edge,theMicron->outer.edge.value,&theMicron->outer.edge.text);
	ext2Str(&formats.edge,theMicron->inner.edge.value,&theMicron->inner.edge.text);
}
/********************************************************************/
ext2Str(theDecForm,theExtended,theString)

	decform		*theDecForm;
	extended	theExtended;
	Str255		*theString;
{
	decimal		d;
	char		s;

	num2dec(theDecForm,theExtended,&d);
	dec2str(theDecForm,&d,&s);
	PtoCstr(theString);
	strcpy(theString,&s);
	CtoPstr(theString);
}
/********************************************************************/
void calcGlobalDistances() {

	extended	deltaX,X,S;

	deltaX = (micron1.inner.edge.value + micron2.inner.edge.value) * 0.5;
	X = (distance_X1 * 1000000 + distance_X2 * 1000) * cfx * 0.5;

	micron1.inner.center.value = X + deltaX;
	micron2.inner.center.value = X + deltaX;
	micron1.outer.center.value = micron1.inner.center.value + micron1.outer.edge.value -
		micron1.inner.edge.value;
	micron2.outer.center.value = micron2.inner.center.value + micron2.outer.edge.value -
		micron2.inner.edge.value;
	ext2Str(&formats.center,micron1.outer.center.value,&micron1.outer.center.text);
	ext2Str(&formats.center,micron2.outer.center.value,&micron2.outer.center.text);
	ext2Str(&formats.center,micron1.inner.center.value,&micron1.inner.center.text);
```

```
          ext2Str(&formats.center,micron2.inner.center.value,&micron2.inner.center.text);

          S = (distance_s1 * 1000000 + distance_s2 * 1000) * cfs;

          micron1.outer.angle.value = atan(micron1.outer.center.value / S);
          micron1.inner.angle.value = atan(micron1.inner.center.value / S);
          micron2.outer.angle.value = atan(micron2.outer.center.value / S);
          micron2.inner.angle.value = atan(micron2.inner.center.value / S);
          ext2Str(&formats.angle,micron1.outer.angle.value,&micron1.outer.angle.text);
          ext2Str(&formats.angle,micron1.inner.angle.value,&micron1.inner.angle.text);
          ext2Str(&formats.angle,micron2.outer.angle.value,&micron2.outer.angle.text);
          ext2Str(&formats.angle,micron2.inner.angle.value,&micron2.inner.angle.text);
}
/***********************************************************************/
void displayDistances()

{
          setText(controlDialog,outerNode1PixelsST,&micron1.outer.pixels.text);
          setText(controlDialog,innerNode1PixelsST,&micron1.inner.pixels.text);
          setText(controlDialog,innerNode2PixelsST,&micron2.inner.pixels.text);
          setText(controlDialog,outerNode2PixelsST,&micron2.outer.pixels.text);

          setText(controlDialog,outerNode1EdgeST,&micron1.outer.edge.text);
          setText(controlDialog,innerNode1EdgeST,&micron1.inner.edge.text);
          setText(controlDialog,innerNode2EdgeST,&micron2.inner.edge.text);
          setText(controlDialog,outerNode2EdgeST,&micron2.outer.edge.text);

          setText(controlDialog,outerNode1CenterST,&micron1.outer.center.text);
          setText(controlDialog,innerNode1CenterST,&micron1.inner.center.text);
          setText(controlDialog,innerNode2CenterST,&micron2.inner.center.text);
          setText(controlDialog,outerNode2CenterST,&micron2.outer.center.text);

          setText(controlDialog,outerNode1ThetaST,&micron1.outer.angle.text);
          setText(controlDialog,innerNode1ThetaST,&micron1.inner.angle.text);
          setText(controlDialog,innerNode2ThetaST,&micron2.inner.angle.text);
          setText(controlDialog,outerNode2ThetaST,&micron2.outer.angle.text);
```

```
}
/*********************************************************************/
void calculateDiameters() {

        displayMessage(message16SRID,nullSRID,nullSRID,nullSRID);

        diameter1 = doCalculateDiameter(&micron1);
        diameter2 = doCalculateDiameter(&micron2);
        diameterAverage = (diameter1 + diameter2) * 0.5;

        ext2Str(&formats.diameter,diameter1,&diameter1String);
        ext2Str(&formats.diameter,diameter2,&diameter2String);
        ext2Str(&formats.diameter,diameterAverage,&diameterAverageString);

        displayDiameters();

        displayMessage(nullSRID,nullSRID,nullSRID,nullSRID);
}
/*********************************************************************/
extended doCalculateDiameter(theMicron)

        micronEye       *theMicron;
{
        extended        ds1,ds2,k1,k2,df1,df2;

        ds1 = nodeNumber * wavelength / sin(theMicron->outer.angle.value);
        ds2 = nodeNumber * wavelength / sin(theMicron->inner.angle.value);

        k1 = coefficientK1(nodeNumber);
        k2 = coefficientK2(nodeNumber);

        df1 = ds1 * (1   k1 * power(ds1 * 0.000001,k2));
        df2 = ds2 * (1 - k1 * power(ds2 * 0.000001,k2));

        return((df1 + df2) * 0.5);
}
```

```
/*****************************************************************/
extended coefficientK1(n)

	extended	n;
{
	extended	result;

	result =	17.7800 - 1.01925 * n - 3.56279 * ipower(n,2);
	result +=	1.31954 * ipower(n,3) - 0.187208 * ipower(n,4);
	result +=	0.00970833 * ipower(n,5);
	return(result * 0.000001);
}
/*****************************************************************/
extended coefficientK2(n)

	extended n;
{
	extended result;

	result =	-0.67780 + 0.0277600 * n - 0.00426917 * ipower(n,2);
	result +=	0.01130 * ipower(n,3) - 0.00155833 * ipower(n,4);
	result +=	0.000090 * ipower(n,5);
	return(result);
}
/*****************************************************************/
void displayDiameters() {

	setText(controlDialog,diameter1ST,&diameter1String);
	setText(controlDialog,diameter2ST,&diameter2String);
	setText(controlDialog,diameter12ST,&diameterAverageString);
}
/*****************************************************************/
void flipVertical(theBitMap)

	BitMap		*theBitMap;
{
	BitMap		tempBitMap;
	Rect		rect1,rect2,rect3,rect4;
	INTEGER		row;

	tempBitMap.baseAddr = NewPtr(8192);
	tempBitMap.rowBytes = 64;
	SetRect(&tempBitMap.bounds,0,0,512,128);

	for(row = 0; row < 128; row++) {
		SetRect(&rect1,0,row,512,row+1);
		SetRect(&rect2,0,127-row,512,128-row);
		CopyBits(theBitMap,&tempBitMap,&rect1,&rect2,srcCopy,NIL);
	}

	SetRect(&rect1,0,0,512,32);
	SetRect(&rect2,0,32,512,64);
	SetRect(&rect3,0,64,512,96);
	SetRect(&rect4,0,96,512,128);
	CopyBits(&tempBitMap,theBitMap,&rect1,&rect1,srcCopy,NIL);
	CopyBits(&tempBitMap,theBitMap,&rect2,&rect2,srcCopy,NIL);
	CopyBits(&tempBitMap,theBitMap,&rect3,&rect3,srcCopy,NIL);
	CopyBits(&tempBitMap,theBitMap,&rect4,&rect4,srcCopy,NIL);

	DisposPtr(tempBitMap.baseAddr);
}
/******************************************************************************/
void digitize() {

	BOOLEAN		show1,show2;

	show1 = micron1.show;
	show2 = micron2.show;

	micron1.show = TRUE;
	micron2.show = TRUE;
```

```
		doScanNow(SLOW,SINGLE);

		micron1.show = show1;
		micron2.show = show2;
		return;
	}
	/********************************************************************/
	void autopilot() {

		EventRecord		theEvent;
		INTEGER			eventMask;

		eventMask = mDownMask + keyDownMask;

		digitize();

		if(!EventAvail(eventMask,&theEvent)) {
			findNodes();
		} else {
			return;
		}

		if(!EventAvail(eventMask,&theEvent)) {
			calculateDiameters();
		}
	}
```

```
* Resource code for CALIPER © 1987.  Used with RMaker.
* Written by Lieutenant Jeffrey S. Kunkel
* Naval Postgraduate School, Monterey, California
*

CALIPER.rsrc                        ;; title of output file
                                    ;; no file type or creator bytes

Type ALRT                                 ;; Alert Template

                                                ;; problem with exposure settings
  ,132                                          ;; resource ID
50 50 150 450                                   ;; top left bottom right
132                                             ;; resource ID of item list
7765                                            ;; stages word in hexadecimal

                                                ;; problem with micron eye camera
  ,133                                          ;; resource ID
50 50 150 450                                   ;; top left bottom right
133                                             ;; resource ID of item list
7665                                            ;; stages word in hexadecimal

Type DITL                                 ;; Dialog Item List

                                                ;; control dialog
  ,128 (4)                                      ;; controlDRID  (preload)
41                                              ;; number of items in the list

*  1                                            ;; MicronEye™ #1 button
radioButton enabled                             ;; radio button dialog item, enabled
4 6 20 129                                      ;; top left bottom right
MicronEye™ #1                                   ;; text

*  2                                            ;; MicronEye™ #2 button
radioButton enabled                             ;; radio button dialog item, enabled
28 6 44 129                                     ;; top left bottom right
```

```
MicronEye™ #2                               ;; text

*  3                                        ;; exposure setting for MicronEye™ #1
editText disabled                           ;; editable text dialog item, disabled
4 139 20 187                                ;; top left bottom right
3.00                                        ;; initial setting

*  4                                        ;; exposure setting for MicronEye™ #2
editText disabled                           ;; editable text dialog item, disabled
28 139 44 187                               ;; top left bottom right
4.00                                        ;; initial setting

*  5                                        ;; label
staticText disabled                         ;; static text dialog item, disabled
4 193 20 253                                ;; top left bottom right
seconds                                     ;; text

*  6                                        ;; label
staticText disabled                         ;; static text dialog item, disabled
28 193 44 253                               ;; top left bottom right
seconds                                     ;; text

*  7                                        ;; label
staticText disabled                         ;; static text dialog item, disabled
86 2 102 91                                 ;; top left bottom right
Outer node 1                                ;; text

*  8                                        ;; label
staticText disabled                         ;; static text dialog item, disabled
103 2 119 91                                ;; top left bottom right
Inner node 1                                ;; text

*  9                                        ;; label
staticText disabled                         ;; static text dialog item, disabled
122 2 138 91                                ;; top left bottom right
Inner node 2                                ;; text
```

```
*  10                                          ;; label
staticText disabled                            ;; static text dialog item, disabled
139 2 155 91                                   ;; top left bottom right
Outer node 2                                   ;; text

* 11                                           ;; label
staticText disabled                            ;; static text dialog item, disabled
70 91 86 136                                   ;; top left bottom right
pixels                                         ;; text

* 12                                           ;; Outer node 1 value
staticText disabled                            ;; static text dialog item, disabled
86 97 102 125                                  ;; top left bottom right
888                                            ;; initial value

* 13                                           ;; Inner node 1 value
staticText disabled                            ;; static text dialog item, disabled
103 97 119 125                                 ;; top left bottom right
888                                            ;; initial value

* 14                                           ;; Inner node 2 value
staticText disabled                            ;; static text dialog item, disabled
122 97 138 125                                 ;; top left bottom right
888                                            ;; initial value

* 15                                           ;; Outer node 2 value
staticText disabled                            ;; static text dialog item, disabled
139 97 155 125                                 ;; top left bottom right
888                                            ;; initial value

* 16                                           ;; label
staticText disabled                            ;; static text dialog item, disabled
52 141 86 185                                  ;; top left bottom right
µm to edge                                     ;; text
```

```
* 17                                    ;; Outer node 1 distance to chip edge
staticText disabled                     ;; static text dialog item, disabled
86 136 102 190                          ;; top left bottom right
88888.8                                 ;; initial value

* 18                                    ;; Inner node 1 distance to chip edge
staticText disabled                     ;; static text dialog item, disabled
103 136 119 190                         ;; top left bottom right
88888.8                                 ;; initial value

* 19                                    ;; Inner node 2 distance to chip edge
staticText disabled                     ;; static text dialog item, disabled
122 136 138 190                         ;; top left bottom right
88888.8                                 ;; initial value

* 20                                    ;; Outer node 2 distance to chip edge
staticText disabled                     ;; static text dialog item, disabled
139 136 155 190                         ;; top left bottom right
88888.8                                 ;; initial value

* 21                                    ;; label
staticText disabled                     ;; static text dialog item, disabled
52 199 86 247                           ;; top left bottom right
 µm to center                           ;; text

* 22                                    ;; Outer node 1 distance to pattern center
staticText disabled                     ;; static text dialog item, disabled
86 196 102 250                          ;; top left bottom right
88888.8                                 ;; initial value

* 23                                    ;; Inner node 1 distance to pattern center
staticText disabled                     ;; static text dialog item, disabled
103 196 119 250                         ;; top left bottom right
88888.8                                 ;; initial value

* 24                                    ;; Inner node 2 distance to pattern center
```

```
staticText disabled                              ;; static text dialog item, disabled
122 196 138 250                                  ;; top left bottom right
88888.8                                          ;; initial value

* 25                                             ;; Outer node 2 distance to pattern center
staticText disabled                              ;; static text dialog item, disabled
139 196 155 250                                  ;; top left bottom right
88888.8                                          ;; initial value

* 26                                             ;; label
staticText disabled                              ;; static text dialog item, disabled
52 256 86 320                                    ;; top left bottom right
  angle (radians)                                ;; text

* 27                                             ;; Outer node 1 angle
staticText disabled                              ;; static text dialog item, disabled
86 256 102 320                                   ;; top left bottom right
88888.8                                          ;; initial value

* 28                                             ;; Inner node 1 angle
staticText disabled                              ;; static text dialog item, disabled
103 256 119 320                                  ;; top left bottom right
88888.8                                          ;; initial value

* 29                                             ;; Inner node 2 angle
staticText disabled                              ;; static text dialog item, disabled
122 256 138 320                                  ;; top left bottom right
88888.8                                          ;; initial value

* 30                                             ;; Outer node 2 angle
staticText disabled                              ;; static text dialog item, disabled
139 256 155 320                                  ;; top left bottom right
88888.8                                          ;; initial value

* 31                                             ;; CALIPER icon
iconItem disabled                                ;; icon item, disabled
```

```
18 401 50 433                       ;; top left bottom right
128                                 ;; CALIPER icon resource ID number

* 32                                ;; label
staticText disabled                 ;; static text dialog item, disabled
68 348 86 487                       ;; top left bottom right
Calculated Diameter                 ;; text

* 33                                ;; label
staticText disabled                 ;; static text dialog item, disabled
93 325 109 429                      ;; top left bottom right
MicronEye™ #1                       ;; text

* 34                                ;; label
staticText disabled                 ;; static text dialog item, disabled
113 325 129 429                     ;; top left bottom right
MicronEye™ #2                       ;; text

* 35                                ;; label
staticText disabled                 ;; static text dialog item, disabled
133 325 149 429                     ;; top left bottom right
average                             ;; text

* 36                                ;; MicronEye™ #1 diameter value
staticText disabled                 ;; static text dialog item, disabled
93 435 109 482                      ;; top left bottom right
88.888                              ;; initial value

* 37                                ;; MicronEye™ #2 diameter value
staticText disabled                 ;; static text dialog item, disabled
113 435 129 482                     ;; top left bottom right
88.888                              ;; initial value

* 38                                ;; MicronEye™ #1 & #2 diameter value
staticText disabled                 ;; static text dialog item, disabled
133 435 149 482                     ;; top left bottom right
```

```
88.888                                          ;; initial value

* 39                                            ;; label
staticText disabled                             ;; static text dialog item, disabled
93 484 109 510                                  ;; top left bottom right
µm                                              ;; text

* 40                                            ;; label
staticText disabled                             ;; static text dialog item, disabled
113 484 129 510                                 ;; top left bottom right
µm                                              ;; text

* 41                                            ;; label
staticText disabled                             ;; static text dialog item, disabled
133 484 149 510                                 ;; top left bottom right
µm                                              ;; text


                                                ;; settings modal dialog
 ,129 (4)                                       ;; setUpDialogRID
29                                              ;; number of items in the list

*  1                                            ;; ok button
button enabled                                  ;; button dialog item, enabled
218 280 238 336                                 ;; top left bottom right
OK                                              ;; message (inside the button)

*  2                                            ;; cancel button
button enabled                                  ;; button dialog item, enabled
218 353 238 409                                 ;; top left bottom right
Cancel                                          ;; message (inside the button)

*  3                                            ;; label
staticText disabled                             ;; static text dialog item, disabled
24 2 40 224                                     ;; top left bottom right
Fiber to MicronEyes™ S (coarse)                 ;; text
```

```
*  4                                   ;; label
staticText disabled                    ;; static text dialog item, disabled
48 2 64 224                            ;; top left bottom right
Fiber to MicronEyes™ s (fine)          ;; text

*  5                                   ;; label
staticText disabled                    ;; static text dialog item, disabled
72 2 88 224                            ;; top left bottom right
Between MicronEyes™ X (coarse)         ;; text

*  6                                   ;; label
staticText disabled                    ;; static text dialog item, disabled
96 2 112 224                           ;; top left bottom right
Between MicronEyes™ x (fine)           ;; text

*  7                                   ;; label
staticText disabled                    ;; static text dialog item, disabled
2 228 18 296                           ;; top left bottom right
Distances                              ;; text

*  8                                   ;; S value
editText disabled                      ;; editable text dialog item, disabled
24 232 40 292                          ;; top left bottom right
0.28                                   ;; initial value

*  9                                   ;; s value
editText disabled                      ;; editable text dialog item, disabled
48 232 64 292                          ;; top left bottom right
0.0                                    ;; initial value

*  10                                  ;; X value
editText disabled                      ;; editable text dialog item, disabled
72 232 83 292                          ;; top left bottom right
0.0                                    ;; initial value
```

```
*  11                                      ;; x value
editText disabled                          ;; editable text dialog item, disabled
96 232 112 292                             ;; top left bottom right
82.5                                       ;; initial value

*  12                                      ;; label
staticText disabled                        ;; static text dialog item, disabled
24 298 40 358                              ;; top left bottom righ
meters                                     ;; text

*  13                                      ;; label
staticText disabled                        ;; static text dialog item, disabled
48 298 64 382                              ;; top left bottom right
millimeters                                ;; text

*  14                                      ;; label
staticText disabled                        ;; static text dialog item, disabled
72 298 88 358                              ;; top left bottom right
meters                                     ;; text

*  15                                      ;; label
staticText disabled                        ;; static text dialog item, disabled
96 298 112 382                             ;; top left bottom right
millimeters                                ;; text

* 16                                       ;; CALIPER icon
iconItem disabled                          ;; icon item, disabled
26 399 90 463                              ;; top left bottom right
128                                        ;; CALIPER icon resource ID number

*  17                                      ;; label
staticText disabled                        ;; static text dialog item, disabled
120 2 136 158                              ;; top left bottom right
Correction factor CFs                      ;; text

*  18                                      ;; label
```

```
staticText disabled                 ;; static text dialog item, disabled
144 2 160 158                       ;; top left bottom right
Correction factor CFx               ;; text

*  19                               ;; label
staticText disabled                 ;; static text dialog item, disabled
168 2 184 158                       ;; top left bottom right
Node number                         ;; text

*  20                               ;; label
staticText disabled                 ;; static text dialog item, disabled
192 2 208 158                       ;; top left bottom right
Laser wavelength                    ;; text

*  21                               ;; CFs value
editText disabled                   ;; editable text dialog item, disabled
120 166 136 234                     ;; top left bottom right
1.0                                 ;; initial value

*  22                               ;; CFx value
editText disabled                   ;; editable text dialog item, disabled
144 166 160 234                     ;; top left bottom right
1.0                                 ;; initial value

*  23                               ;; Node number value
editText disabled                   ;; editable text dialog item, disabled
168 166 184 186                     ;; top left bottom right
2                                   ;; initial value

*  24                               ;; Laser wavelength value
editText disabled                   ;; editable text dialog item, disabled
192 166 208 234                     ;; top left bottom right
0.6328                              ;; initial value

*  25                               ;; label
staticText disabled                 ;; static text dialog item, disabled
```

```
192 240 208 272                             ;; top left bottom right
µm                                          ;; text

*  26                                       ;; label
staticText disabled                         ;; static text dialog item, disabled
216 2 232 158                               ;; tcp left bottom right
Positive squelch level                      ;; text

*  27                                       ;; label
staticText disabled                         ;; static text dialog item, disabled
240 2 256 158                               ;; top left bottom right
Negative squelch level                      ;; text

*  28                                       ;; label
editText disabled                           ;; edit text dialog item, disabled
216 166 232 234                             ;; top left bottom right
0.08                                        ;; initial value

*  29                                       ;; label
editText disabled                           ;; edit text dialog item, disabled
240 166 256 234                             ;; top left bottom right
0.15                                        ;; initial value


                                            ;; message modeless dialog
  ,131                                      ;; messageDialogRID
1                                           ;; number of items in the list

*  1                                        ;; message static text item
staticText disabled                         ;; static text dialog item, disabled
0 0 16 510                                  ;; top left bottom right
^0 ^1 ^2 ^3                                 ;; message text

                                            ;; alert box, one button
  ,132 (4)                                  ;; messageDialogRID
3                                           ;; number of items in the list
```

```
*  1                                       ;; message static text item
button enabled                             ;; static text dialog item, disabled
6 320 26 390                               ;; top left bottom right
Abort                                      ;; message text

*  2                                       ;; message static text item
staticText disabled                        ;; static text dialog item, disabled
3 50 96 307                                ;; top left bottom right
^0                                         ;; message text

*  3
iconItem disabled                          ;; icon item, disabled
3 9 35 41                                  ;; top left bottom right
0                                          ;; stop icon

                                           ;; alert box, two buttons
  ,133 (4)                                 ;; messageDialogRID
4                                          ;; number of items in the list

*  1                                       ;; message static text item
button enabled                             ;; static text dialog item, disabled
6 320 26 390                               ;; top left bottom right
Abort                                      ;; message text

*  2                                       ;; message static text item
button enabled                             ;; static text dialog item, disabled
35 320 55 390                              ;; top left bottom right
Continue                                   ;; message text

*  3                                       ;; message static text item
staticText disabled                        ;; static text dialog item, disabled
3 50 96 307                                ;; top left bottom right
^0 ^1                                      ;; message text

*  4
```

```
iconItem disabled                                  ;; icon item, disabled
3 9 35 41                                          ;; top left bottom right
0                                                  ;; stop icon

                                                   ;; error dialog
   ,9999 (4)                                       ;; errDialogRID
6
*  1
button enabled
166 171 186 227
OK

*  2
staticText disabled
40 8 56 388
^1

*  3
staticText disabled
64 8 80 388
^2

*  4
staticText disabled
88 8 154 388
^3

*  5
staticText disabled
11 47 27 388
^0

*  6
iconItem disabled                                  ;; icon item, disabled
3 9 35 41                                          ;; top left bottom right
0                                                  ;; stop icon
```

```
Type DLOG                                   ;; Dialog Template

                                                   ;; control modeless dialog
   ,128                                            ;; resource ID
Control Modeless Dialog                            ;; message (not used)
185 0 341 512                                      ;; top left bottom right
Invisible NoGoAway                                 ;; box status
2                                                  ;; dialog definition ID (plain box)
0                                                  ;; reference value (not used)
128                                                ;; ID of item list ('DITL', above)

                                                   ;; setUp modal dialog
   ,129                                            ;; resource ID
Settings Modal Dialog                              ;; message (not used)
47 16 310 496                                      ;; top left bottom right
Invisible NoGoAway                                 ;; box status
1                                                  ;; dialog definition ID (modal dialog box)
0                                                  ;; reference value (not used)
129                                                ;; ID of item list ('DITL', above)

                                                   ;; message modeless dialog
   ,131                                            ;; resource ID
Message Modeless Dialog                            ;; message (not used)
168 0 183 512                                      ;; top left bottom right
Invisible NoGoAway                                 ;; box status
2                                                  ;; dialog definition ID (plain box)
0                                                  ;; reference value (not used)
131                                                ;; ID of item list ('DITL', above)

                                                   ;; error message dialog
    ,9999                                          ;; resource ID
A Message...                                       ;; message (not used)
87 67 285 478                                      ;; top left bottom right
Invisible NoGoAway                                 ;; box status
```

```
1                                                         ;; dialog definition ID
0                                                         ;; reference value (not used)
9999                                                      ;; ID of item list ('DITL', above)

Type ICN# = GNRL                                    ;; Icon Lists (icon and its mask)

 ,128                                                     ;; resource ID for CALIPER icon
.H                                                        ;; hexadecimal data to follow
0001 C000 003F C000 0011 C000 0089 E000                   ;; icon and icon mask data
00C5 D000 00A3 C820 0091 C450 0089 C2C8
0085 C124 0083 E292 0041 D601 0020 C902
0010 5484 0008 3008 0004 4810 0002 A420
0001 8040 0002 4080 0005 2100 000C 0200
0012 C400 0029 0800 0060 1000 0090 2000
0148 4000 0300 C000 0481 C000 0243 C000
0105 C000 0089 C000 0051 C000 0021 C000
0001 C000 003F C000 001F C000 008F E000
00C7 F000 00E3 F820 00F1 FC70 00F9 FEF8
00FD FFFC 00FF FFFE 007F DFFF 003F CFFE
001F DFFC 000F FFF8 0007 FFF0 0003 FFE0
0001 FFC0 0003 FF80 0007 FF00 000F FE00
001F FC00 003F F800 007F F000 00FF E000
01FF C000 03FF C000 07FF C000 03FF C000
01FD C000 00F9 C000 0071 C000 0021 C000

Type ICON = GNRL                                    ;; Icons

 ,128                                                     ;; resource ID for CALIPER icon
.H                                                        ;; hexadecimal data to follow
0001 C000 003F C000 0011 C000 0089 E000                   ;; icon data
00C5 D000 00A3 C820 0091 C450 0089 C2C8
0085 C124 0083 E292 0041 D601 0020 C902
0010 5484 0008 3008 0004 4810 0002 A420
0001 8040 0002 4080 0005 2100 000C 0200
0012 C400 0029 0800 0060 1000 0090 2000
0148 4000 0300 C000 0481 C000 0243 C000
```

```
0105 C000 0089 C000 0051 C000 0021 C000

Type MENU                       ;; Menu Resource

      ,1                            ;; appleMRID
\14                                 ;; menu title (apple icon)
                                    ;; Must be followed by a blank line

      ,2                            ;; fileMRID
File                                ;; menu title
Quit/Q                              ;; item 1 command key equivalent Q
                                    ;; Must be followed by a blank line

      ,3                            ;; commandsMRID
Commands                            ;; menu title
Equipment Setup...                  ;; setupMI
(-                                  ;; item 2 (draw a line)
Digitize/D                          ;; digitizeMI
Find Nodes/F                        ;; findNodesMI
Calculate Diameters/=               ;; calculateDiametersMI
(-                                  ;; item 6 (draw a line)
Autopilot/A                         ;; automaticMI
                                    ;; Must be followed by a blank line

      ,4                            ;; scanMRID
Scan                                ;; menu title
Single Slow Scan                    ;; singleSlowScanMI
Single Fast Scan                    ;; singleFastScanMI
(-                                  ;; item 3 (draw a line)
Multiple Slow Scans                 ;; multipleSlowScansMI
Multiple Fast Scans                 ;; multipleFastScansMI
(-                                  ;; item 6 (draw a line)
Show MicronEye™ #1                  ;; showMicron1MI
Show MicronEye™ #2                  ;; showMicron2MI
                                    ;; Must be followed by a blank line
```

```
Type STR                                ;; String Resources

  ,128                                      ;; nullSRID
\20                                         ;; ASCII code for blank space

  ,129                                      ;; micronEye1SRID
Micron Eye #1                               ;; title of micron eye #1

  ,130                                      ;; micronEye2SRID
Micron Eye #2                               ;; title of micron eye #2

  ,131                                      ;; message1SRID
  getting exposure settings                 ;; message #1 text

  ,132                                      ;; message2SRID
  initializing                              ;; message #2 text

  ,133                                      ;; message3SRID
  scanning                                  ;; message #3 text

  ,134                                      ;; message4SRID
 is not responding ++
correctly.  Ensure that ++
the camera is connected ++
to the correct port (printer ++
or modem).

  ,135                                      ;; message5RID
  correcting                                ;; message #5 text

  ,136                                      ;; message6SRID
One of the exposure settings contains an ++
illegal character.  The exposure settings must ++
all be between zero and 1000.
```

```
  ,137                                         ;; message7SRID
There is still a problem ++
with at least one of the ++
exposure settings.  Try ++
entering all of the settings ++
again.

  ,138                                         ;; message8SRID
The exposure settings are ++
still invalid.  Enter all ++
of the exposure settings ++
again.  \0D++
0 <= exposure setting <= 1000

  ,139                                         ;; message9SRID
The exposure settings are ++
still invalid.

  ,140                                         ;; message10SRID
One of the geometry inputs is ++
either out of limits or ++
contains an invalid character.++

  ,141                                         ;; message11SRID
There is still some ++
difficulty with one or ++
more of the geometry inputs.++
Try entering all of the ++
inputs again.

  ,142                                         ;; message12SRID
At least one of the inputs ++
is still either negative or ++
too large, or it contains an ++
illegal character.
```

```
  ,143                                          ;; message13SRID
At least one of the inputs ++
is still wrong.

  ,144                                          ;; message14SRID
  finding nodes for

  ,145                                          ;; message15SRID
  calculating distances

  ,146                                          ;; message16SRID
  calculating diameters


Type WIND                                   ;; Window Template

  ,128                                          ;; imageWRID
Micron Eye Scan Output                          ;; title (not used)
38 0 156 512                                    ;; top left bottom right
Invisible NoGoAway                              ;; window status
4                                               ;; window definition ID (document window w/o size box)
0                                               ;; reference value (not used)
```

LIST OF REFERENCES

1. Harlow, D. Gary, and S. Leigh Phoenix. 1978. The Chain-of-Bundles Probability Model for the Strength of Fibrous Materials I: Analysis and Conjectures. *J. Composite Materials* 12 (Apr.): 195-214.

2. Harlow, D. Gary, and S. Leigh Phoenix. 1978. The Chain-of-Bundles Probability Model for the Strength of Fibrous Materials II: A Numerical Study of Convergence. *J. Composite Materials* 12 (Jul.): 314-34.

3. Phoenix, S. L., and E. M. Wu. 1983. Statistics for the Time Dependent Failure of Kevlar-49/Epoxy Composites: Micromechanical Modeling and Data Interpretation. *Mechanics of Composite Materials: Recent Advances*, ed. Zvi Hashin and Carl T. Herakovich. New York: Pergamon Press, 135-62.

4. Rosen, B. Walter. 1964. Tensile Failure of Fibrous Composites. *AIAA Journal* 2 (Nov.): 1985-91.

5. Phoenix, S. L., and R. L. Smith. 1983. A Comparison of Probabilistic Techniques for the Strength of Fibrous Materials Under Local Load-Sharing Among Fibers. *Int. J. Solids Structures* 19, no. 6: 479-496.

6. Phoenix, S. L. 1986. Fiber Bundles: Strength Statistics. *Encyclopedia of Materials Science and Engineering*. ed. Michael B. Bever. Oxford: Pergamon Press. III, 1707-11.

7. Weibull, Waloddi. 1951. A Statistical Distribution Function of Wide Applicability. *Journal of Applied Mechanics* (Sep.): 293-7.

8. Bury, Karl V. 1975. *Statistical Models in Applied Science*. Wiley Series in Probability and Mathematical Statistics. New York: John Wiley & Sons.

9. Harlow, D. Gary, and S. Leigh Phoenix. 1979. Bounds on the Probability of Failure of Composite Materials. *Int. Journ. of Fracture* 15: 321-36.

10. Harlow, D. Gary, and S. Leigh Phoenix. 1981. Probability Distributions for the Strength of Composite Materials II: A Convergent Sequence of Tight Bounds. *Int. Journ. of Fracture* 17: 601-30.

11. Harlow, D. Gary, and S. Leigh Phoenix. 1982. Probability Distributions for the Strength of Fibrous Materials Under Local Load Sharing I: Two-Level Failure and Edge Effects. *Adv. Appl. Prob.* 14: 68-94.

12. Bennett, Thomas Alvin. 1985. *A Comparison of Two Methods for Fiber Diameter Measurement and a System Design for the Study of Composite Reliability*. M.S.A.E. Thesis, Naval Postgraduate School, Monterey, California, (Dec.).

13. Micron Technology Inc. *MicronEye Operator's Manual*. Boise, Idaho.

14. Storch, Mark Gerald. 1986. *A Computer Aided Method for the Measurement of Fiber Diameters by Laser Diffraction*. M.S.A.E. Thesis, Naval Postgraduate School, Monterey, California, (Sep.).

15. Kerker, Milton. 1969. *The Scattering of Light and Other Electromagnetic Radiation*. New York: Academic Press.

16. Tipler, Paul A. 1976. *Physics*. New York: Worth Publishers, Inc.

17. Perry, A. J., B. Ineichen, and B. Eliasson. 1974. Fibre Diameter Measurement by Laser Diffraction. *Journal of Materials Science*. 9: 1376-8.

18. CAMERA version 1.1. 1984. Mesa Graphics, Los Alamos, New Mexico.

19. Cummings, Bryan J., and Lawrence J. Pollack. 1986. *Programming The Macintosh™ In* C. Berkeley: SYBEX.

20. Kernighan, Brian W., and Dennis M. Ritchie. 1978. *The C Programming Language*. Englewood Cliffs, N.J.: Prentice-Hall.

21. *Apple® Numerics Manual*. 1986. Reading, Mass.: Addison-Wesley.

22. *Inside Macintosh™*. 1985. Vols. I, II, III. Reading, Mass.: Addison-Wesley.

BIBLIOGRAPHY

Abramowitz, Milton, and Irene A. Stegun. "Generation of Bessel Functions on High Speed Computers." In *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables.* National Bureau of Standards Applied Mathematics Series. Washington, D. C.: U. S. Government Printing Office, 1972.

Apple® Numerics Manual. Reading, Mass.: Addison-Wesley, 1986.

Bennett, Thomas Alvin. *A Comparison of Two Methods for Fiber Diameter Measurement and a System Design for the Study of Composite Reliability.* M.S.A.E. Thesis, Naval Postgraduate School, Monterey, California, Dec. 1985.

Bury, Karl V. *Statistical Models in Applied Science.* Wiley Series in Probability and Mathematical Statistics. New York: John Wiley & Sons, 1975.

CAMERA version 1.1, Mesa Graphics, Los Alamos, New Mexico, 1984.

Chamis, C. C. "Micromechanics Strength Theories." In *Composite Materials.* Vol. 5, *Fracture and Fatigue,* ed. Lawrence J. Broutman. New York: Academic Press, 1974, 93-151.

Chernicoff, Stephen. *Macintosh™ Revealed.* Vol. 1, *Unlocking the Toolbox.* Hasbrouck Heights, N. J.: Hayden Book Company, 1985.

————. *Macintosh™ Revealed.* Vol. 2, *Programming with the Toolbox.* Hasbrouck Heights, N. J.: Hayden Book Company, 1985.

Chistova, E. A. *Tables of Bessel Functions of the True Argument and of Integrals Derived From Them.* London: Pergamon Press, 1959.

Cummings, Bryan J., and Lawrence J. Pollack. *Programming The Macintosh™ In* C. Berkeley: SYBEX, 1986.

Francis, George C., and Viola Woodward. *BRL Report No. 1197: Tables of Ordinary Bessel Functions of the Second Kind of Orders 0 through 9.* 5 vols. Aberdeen Proving Ground, Maryland: Ballistic Research Laboratories, 1963.

Goldstein, M., and R. M. Thaler. "Recurrence Techniques for the Calculation of Bessel Functions." *Mathematical Tables and other Aids to Computation* 60 (Oct. 1957): 102-8.

Harlow, D. Gary, and S. Leigh Phoenix. "The Chain-of-Bundles Probability Model for the Strength of Fibrous Materials I: Analysis and Conjectures." *J. Composite Materials* 12 (Apr. 1978): 195-214.

————. "The Chain-of-Bundles Probability Model for the Strength of Fibrous Materials II: A Numerical Study of Convergence." *J. Composite Materials* 12 (Jul. 1978): 314-34.

————. "Bounds on the Probability of Failure of Composite Materials." *Int. Journ. of Fracture* 15 (1979): 321-36.

————. "Probability Distributions for the Strength of Composite Materials II: A Convergent Sequence of Tight Bounds." *Int. Journ. of Fracture* 17 (1981): 601-30.

————. "Probability Distributions for the Strength of Fibrous Materials Under Local Load Sharing I: Two-Level Failure and Edge Effects." *Adv. Appl. Prob.* 14 (1982): 68-94.

Huxham, Fred A., David Burnard, and Jim Takatsuka. *Using the Macintosh™ Toolbox with C.* Berkeley: SYBEX, 1986.

Inside Macintosh™. Vols. I, II, III. Reading, Mass.: Addison-Wesley, 1985.

Jones, Robert M. *Mechanics of Composite Materials.* New York: Hemisphere Publishing Corporation, 1975.

Kerker, Milton. *The Scattering of Light and Other Electromagnetic Radiation.* New York: Academic Press, 1969.

Kernighan, Brian W., and Dennis M. Ritchie. *The C Programming Language.* Englewood Cliffs, N.J.: Prentice-Hall, 1978.

MAC C version 5.1. Consulair Corporation, Portola Valley, Calif., 1987.

Marcuse, Dietrich. *Principles of Optical Fiber Measurements.* New York: Academic Press, 1981.

Metcalfe, A. G., and G. K. Schmitz. "Effect of Length on the Strength of Glass Fibers." *American Society for Testing and Materials Proceedings* 64 (1964): 1075-93.

Micron Technology Inc. *MicronEye Operator's Manual.* Boise, Idaho.

Moreton, R. "The Effect of Gauge Length of the Tensile Strength of R.A.E. Carbon Fibres." *Fibre Science and Technology*. Great Britain: Elsevier Publishing Company Ltd. I, 273-84.

Peatroy, David B. *Mastering The Macintosh™ Toolbox*. Berkeley: Osborne McGraw-Hill, 1986.

Perry, A. J., B. Ineichen, and B. Eliasson. "Fibre Diameter Measurement by Laser Diffraction." *Journal of Materials Science*. 9 (1974): 1376-8.

Phoenix, S. L. "Fiber Bundles: Strength Statistics." *Encyclopedia of Materials Science and Engineering*. Ed. Michael B. Bever. Oxford: Pergamon Press, 1986. III, 1707-11.

Phoenix, S. L., and R. L. Smith. "A Comparison of Probabilistic Techniques for the Strength of Fibrous Materials Under Local Load-Sharing Among Fibers." *Int. J. Solids Structures* 19, no. 6 (1983): 479-496.

Phoenix, S. L., and E. M. Wu. "Statistics for the Time Dependent Failure of Kevlar-49/Epoxy Composites: Micromechanical Modeling and Data Interpretation." *Mechanics of Composite Materials: Recent Advances*. Ed. Zvi Hashin and Carl T. Herakovich. New York: Pergamon Press, 1983, 135-62.

Pitt, R. E., and S. L. Phoenix. "Probability Distributions for the Strength of Composite Materials III: The Effect of Fiber Arrangement." *Int. Journ. of Fracture* 20 (1982): 291-311.

Rosen, B. Walter. "Tensile Failure of Fibrous Composites." *AIAA Journal*. 2 (Nov. 1964): 1985-91.

Simpson, Henry. *Programming the Macintosh™ User Interface*. New York: McGraw-Hill, 1986.

The Staff of the Computation Laboratory. *The Annals of the Computation Laboratory of Harvard University*. Vol. 3, *Tables of the Bessel Functions of the First Kind of Orders Zero and One*. Cambridge, Mass.: Harvard University Press, 1947.

Spiegel, Murray R. *Schaum's Outline of Theory and Problems of Probability and Statistics*. Schaum's Outline Series. New York: McGraw-Hill Book Company, 1975

————. *Schaum's Outline of Theory and Problems of Statistics*. Schaum's Outline Series. New York: McGraw-Hill Book Company, 1961.

Storch, Mark Gerald. *A Computer Aided Method for the Measurement of Fiber Diameters by Laser Diffraction.* M.S.A.E. Thesis, Naval Postgraduate School, Monterey, California, Sep. 1986.

Tipler, Paul A. *Physics.* New York: Worth Publishers, Inc., 1976.

Tsai, Stephen W. *Composites Design,* 3rd ed. Dayton: Think Composites, 1987.

Wagner, H. Daniel, S. Leigh Phoenix, and Peter Schwartz. "A Study of Statistical Variability in the Strength of Single Aramid Filaments." *Journal of Composite Materials* 18 (Jul. 1984): 312-38.

Ward, Terry A. *Programming C on the Macintosh.* Glenview, Illinois: Scott, Foresman and Company, 1986.

Weibull, Waloddi. "A Statistical Distribution Function of Wide Applicability." *Journal of Applied Mechanics* (Sep. 1951): 293-7.

Wu, Edward M. "Phenomenological Anisotropic Failure Criterion." In *Composite Materials.* Vol. 2, *Mechanics of Composite Materials,* ed. G. P. Sendeckyj. New York: Academic Press, 1974, 353-431.

Zweben, C. "Fibrous Composites: Themomechanical Properties." *Encyclopedia of Materials Science and Engineering.* Ed. Michael B. Bever. Oxford: Pergamon Press, 1986. III, 1733-41.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Commander, Naval Air Systems Command Assistant Commander for Systems & Engineering (NAIR-05) 1421 Jefferson Davis Highway (JP-2) Arlington, VA 22202	1
2.	Dr. Robert Badaliance Chief, Mechanics of Materials Branch Code 6380 Naval Research Laboratory Washington, D.C. 20375	1
3.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
4.	Superintendent Attn: Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
5.	Dr. Edward M. Wu Professor of Aeronautics, Code 67Wt Naval Postgraduate School Monterey, California 93943-5000	19
6.	Jeffrey S. Kunkel, LT, USN 5114 Piney Branch Road Fairfax, Virginia 22030	5